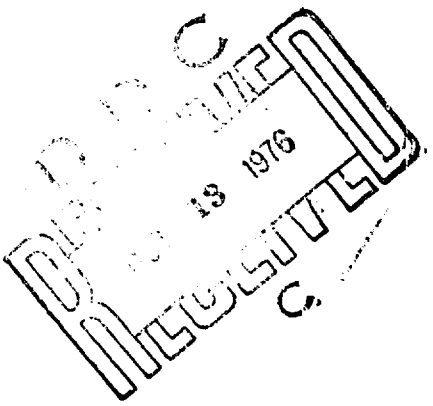
Technical Report No. TR-3534
August 1976

AD-A030 671

ANALYSIS OF RECORD BLOCK CHAINS FOR THE CDC 6700

Alfred H. Morris, Jr.

Warfare Analysis Department



Approved for public release; distribution unlimited

FOREWORD

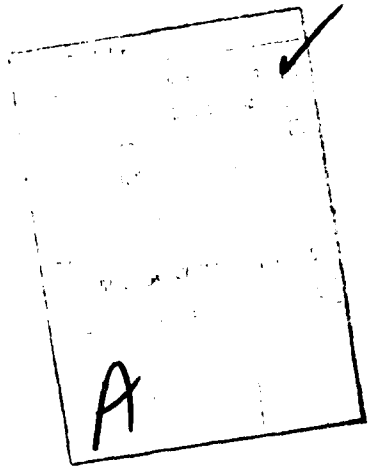
The CDC 6700 computer is a permanent file oriented system. Since system malfunctions can occur, a program should be available for determining if the permanent files are properly stored and catalogued. This report gives a detailed description of a program that handles this task. The program is for the SCOPE 3.4.3 operating system.

The work reported here was funded by the computer systems support code DK9933A. The report was reviewed by Mr. Alban P. Gass and Mr. John G. Perry, Jr. of NSWC/DL. The author is particularly grateful to Mr. Perry for his many suggestions and comments.

Released by:

Ralph A. Niemann

RALPH A. NIEMANN, Head
Warfare Analysis Department



ABSTRACT

This report gives a detailed description of a program, currently in operation at NSWC/DL, that determines if the permanent files are properly stored and catalogued on the CDC 6700 computer. The program serves only in a diagnostic capacity. Its purpose is not to correct errors, but only to find and report them. Such a program is needed since system malfunctions can occur. The documentation is intended primarily for system analysts. The report defines each task that is involved, and examines the coding that performs the task. Background information is provided when needed. The program is for the SCOPE 3.4.3 operating system.

TABLE OF CONTENTS

Introduction

The CDC 6700 is a permanent file oriented machine. A file that is stored on disk will reside in one or more record blocks (RB's). These RB's will be linked to one another, forming what is called an RB chain. The arrangement maintained for referencing the files is fairly comprehensive. Each file is required to have a name, a number (called the cycle number), and an ID. The ID is a word that identifies the creator, owner, or user of the file. Several files may have the same file name. However, files that have the same file name and the same ID must contain different cycle numbers. The Permanent File Catalog (PFC) contains an entry for each file. Included in this entry is the name of the file, its ID, cycle number, creation date, a bit which indicates whether it is stored on tape or disk, and its RB chain (if it is stored on disk). A second file, the Permanent File Directory (PFD), provides the proper facilities for locating the PFC entry for a file. The PFD contains an entry for each file name. An entry contains pointers to all the PFC entries that have the same file name and same ID but different cycle numbers.

Since system malfunctions can occur, a program should be available for determining if the permanent files are properly stored and catalogued. The purpose of this report is to give a detailed description of the RBTC program. This program checks the RB chains for the PFD and PFC, and examines the PFC entries (including their RB chains) for the permanent files. This RB chain/PFC analyzer is for the Scope 3.4.3 operating system.

Whenever the analyzer is used, it is required that no new programs be read into the input queue, no programs in the output queue be printed, and that the program be permitted to run to completion without interruption. In other words, it is required that the patient's condition not be modified until diagnosis is complete. This, however, should cause no hardship. The analyzer can examine the patient quite rapidly. At the present time it takes only 10-15 CDC 6400 seconds for the program to examine the 18000 files on the CDC 6700 at NSWC/DL. Unless many irregularities have to be reported on, the entire job (including printing) normally takes less than 20 CDC 6400 seconds. In contrast, the auditor AUDIT (a CDC program) takes 30 minutes!

The analyzer serves only in a diagnostic capacity. Its purpose is not to correct errors, but only to find and report them. If the files are inadvertently modified while being examined, then no damage is done. This can only cause the program to abort prematurely or to obtain incorrect results.

The administrative structure of the analyzer is quite simple. Each major task is handled by a separate subroutine. The main program calls these routines in sequence. After the routines have finished, the final results are printed and the program terminates. In this report a separate section is devoted to each of the major routines. The routines are discussed in the order that they are called by the main program. Each routine is described in detail. Additional sections are included that provide background information and describe subroutines that are used by the major routines. Listings of the main program, all the major routines, and most of the subsidiary routines are given in appendix A.

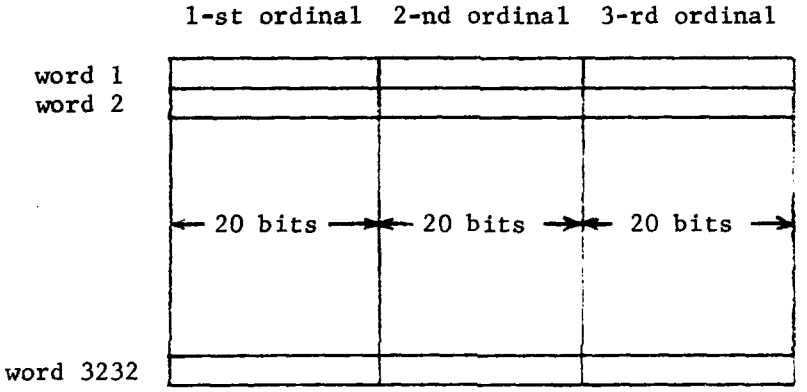
The analysis performed by the analyzer involves the following steps. First the RB chains for the PFD and PFC are examined. If an error is detected in either of the chains, then in most cases the job terminates. The error is considered to be too severe to permit analysis to continue. Otherwise, if no such error is detected, then the PFC entries for all the permanent files are next read and examined. At the time this is done, normally no errors are reported. A temporary file (tape 2) is maintained for referencing the errors. After the entries have been examined, the first half of the program is finished. Now the system must read tape 2, collect together all the errors for each individual file, and report them to the user. The REPORT routine is responsible for coordinating most of these activities. Portions of the PFC may have to be read a second time in order that REPORT can provide all the information that is needed. After REPORT is finished, the results are printed and the program terminates.

At the present time, possibly a maximum of 65000 files can be examined. The precise restrictions on the size and arrangement of the permanent files are itemized on page A-1, at the beginning of the listing of the main program. Several of the restrictions are needed by SETUP, which is the first routine that is called. This routine defines the environment in which the analyzer operates.

The RBRTBL Array and Related Concepts

The major task of the analyzer is to check the RB chains of the permanent files for conflicts. In order to know which RB's are used and where they are used, an array named RBRTBL is used that is similar to the RBR bit tables employed by the system. However, instead of referencing an RB by a single bit, as is done in the RBR bit tables, 20 bits are used in the RBRTBL.

The array RBRTBL is a matrix having 3232 rows and 8 columns. Each column contains 20 bit fields for the RB's of three RBR ordinals, being arranged as follows:



Specifically, bits 40-59 of the i-th word of the column is the 20 bit field for the i-th RB of the first RBR ordinal, bits 20-39 of the i-th word is the 20 bit field for the i-th RB of the second RBR ordinal, and bits 0-19 of the i-th word is the 20 bit field for the i-th RB of the third RBR ordinal.

At the present time RBR ordinals 0-39 are permitted by the analyzer to be used for permanent files. However, since the RBRTBL has 8 columns and each column contains information for the RB's of three RBR ordinals, only 24 of these 40 RBR ordinals can be used at any one time. The 40 RBR ordinals are linked to the RBRTBL by the arrays MORD and NORD, each of which is of dimension 40. If RBR ordinal ORD is being used then MORD(ORD+1) = M and NORD(ORD+1) = N where

N = the number of the column of RBRTBL that contains the ordinal ORD
M = 1,2,3 depending on whether the RBR ordinal ORD is the first, second, or third RBR ordinal of the column.

M and N are called the coordinates of the ordinal ORD in the RBRTBL. If the RBR ordinal ORD is not being used then these coordinates (i.e., MORD(ORD+1) and NORD(ORD+1)) are set to 0.

Consider now the bits in the 20 bit field for an RB. The first bit (on the left) is currently unused and will have the value 0. The second bit is called the flaw bit. If this bit is 1 then the RB is flawed. Otherwise, if the bit is 0

then the RB is unflawed. The remaining 18 bits have one of the three following formats:

(1) If the RB is first found in word n of the RB chain for the PFD then the third bit (called the PFD bit) is set to 1, the fourth bit (called the PFC bit) is set to 0, and the remaining 16 bits contain the number n.
(2) If the RB is first found in word n of the RB chain for the PFC then the PFD bit is set to 0, the PFC bit is set to 1, and the remaining 16 bits contain the number n.
(3) Otherwise, if the RB is first found in an RB chain for a permanent file in the RBTC, then the PFD and PFC bits are set to 0 and the last 16 bits contain the number of the pru in the RBTC where the RB is found.

Here the word and pru counts begin at 0. Hence, neither the PFD nor the PFC can have an RB chain that contains more than 2^{16} words, and the PFC cannot be longer than 2^{16} prus (i.e., 1170 RB's) in length.

The restrictions that the RBTC is no longer than 1170 RB's in length and that at any one time no more than 24 RBR ordinals are being used must be regarded as being permanent. However, the restriction that only 40 RBR ordinals are available for use can easily be removed.

SUBROUTINE SETUP(ERR)

The SETUP routine is responsible for

(1) finding the RBR ordinals for the permanent file devices that are turned on,
(2) setting up the RBRTBL array for these RBR ordinals, and
(3) setting the flaw bits in the RBRTBL for the flawed RB's.

ERR is a variable that is set by the routine to inform the main program if these tasks are completed properly. If an error is detected which requires that the job be aborted prematurely, then an error message is printed and ERR is assigned a nonzero value. Otherwise, if no such error is detected, then ERR is assigned the value 0.

The subroutines

```
CALL PFEST(NUM,LEST)
CALL RLABEL(e,LABEL,ERR)
```

are used by SETUP for finding the RBR ordinals and flawed RB's. Here NUM is a variable and LEST an array. When PFEST is called, NUM and LEST are assigned the values:

NUM = the number of permanent file devices that are turned on
LEST = the list of EST ordinals for the devices that are turned on

If NUM = 0 or NUM > 24 then an error message is printed, the error indicator ERR is set to 1, and SETUP terminates.

If $1 \leq NUM \leq 24$ then for each EST ordinal e in LEST the RLABEL subroutine is called for reading the label of the device.[1] Here LABEL is an array that is 128 words in length. If the label cannot be read then ERR is set by RLABEL to a nonzero value, an error message is printed by SETUP, and SETUP terminates. Otherwise, if the label can be read, then the first 10 words of the label are stored in words 1-10 of LABEL, and words 11-64 of LABEL contain a bit table for the RB's on the device. If a bit is 1 then the corresponding RB is flawed. Otherwise, the RB is unflawed.

Three fields in the first 10 words of LABEL are of interest. The third byte of word 10 contains the EST ordinal for the device. Either this number is the same as the EST ordinal e in LEST, or we have an error. An error in this case indicates that the permanent file pack is not mounted on the correct EST. If such an error occurs then an error message is printed, ERR is set to 1, and SETUP terminates. Secondly, bits 0-5 of word 5 contain the RBR ordinal for this device (here it is assumed that there is only one RBR ordinal for each permanent file device). If ORD is the RBR ordinal and ORD > NUMTBL (NUMTBL is the largest RBR ordinal that can be used), then again we have an error, an error message is printed, ERR is set to 1, and SETUP terminates. Thirdly, the last byte of word 10 contains the number of RB's on the device (both flawed and unflawed). Here it is

[1] For information on labels see pages II-1-61 and II-3-19 of reference (1).

checked that this number, denoted by MAX, satisfies $1 \leq MAX \leq 3232$.

The arrays MORD and NORD, both of dimension 40, are defined next by SETUP. These routines contain the coordinates (M,N) of the RBR ordinals for the permanent file devices. After MORD and NORD are defined, tasks (1) and (2) are finished. Task (3) is now accomplished by the DO loop immediately following statement 31.

Comments. The purpose of the SETUP routine is to set up the RBRTBL, AREA, MORD, and NORD arrays, which define the environment in which the program will operate. The array AREA is not used in SETUP, but is employed in later routines for obtaining the 20 bit fields for the RB's in the RBRTBL. The LEST, LABEL, and FLAW arrays are of interest only in SETUP. They are not used elsewhere. The subroutines PFEST and RLABEL (which are also used only in SETUP) are our interface with the system tables and the label handling operations. One may either use the PFEST and RLABEL routines employed at NSWC/DL (which have separate documentation), or one may use alternate procedures for giving the RBR ordinals and flawed RB's. One may, of course, totally rewrite the SETUP routine. If this is done, then the following two requirements must be met:

(1) The RBRTBL, AREA, MORD, and NORD arrays must still contain the same information. The only exception is that the flaw bits in the RBRTBL need not be set. However, if a flaw bit for an RB is not set, then the program will not know that this RB is flawed.

(2) The error indicator ERR must still be used to inform the main program that an error has occurred which requires that the job be aborted prematurely. Set $ERR \neq 0$ if such an error occurs. Otherwise, set $ERR = 0$.

The assumption that there is only one RBR ordinal for each permanent file device is used only in SETUP. Thus if SETUP is rewritten then this restriction may be removed.

In SETUP it is currently assumed that only 40 RBR ordinals are available for use. The value NUMTBL, which is the largest ordinal that may be used, is assigned the default value 39 in the main program. The restriction on the number of available RBR ordinals can be easily modified by changing the DO statement preceding statement 11, changing the FORMAT statement 121, and replacing the dimensions on the COMMON card:

```
COMMON/RBRORD/MORD(40),NORD(40)
```

This card must also be replaced in the routines RBSET and RBTCHK, and the default value for NUMTBL in the main program should be modified.

Reading the PFD and PFC

The subroutines

```
CALL GETFILE
CALL TEST
CALL PFRETRN
```

permit us to read the PFD and PFC. A routine that uses any of these three subroutines requires the statement

```
COMMON/IFET/IBUFF(2049),RTCODE,NWDS,LEVEL,FILE
```

where RTCODE and FILE are integer variables. The variable FILE is set by the user and may take the values 0 or 1. If FILE = 0 then the PFD is the file to be read. Otherwise, if FILE = 1 then the PFC is the file to be read.

The GETFILE routine is the routine that, in effect, opens the file, TEST is the routine for reading this file after it has been opened, and PFRETRN is the routine for closing the file. More precisely, GETFILE creates a FNT entry for the file requested. After the FNT entry is created, the TEST routine may be called one or more times to read portions of the file. (The first time that TEST is called, a FET for the file is created by TEST.) After the reading has been completed the routine PFRETRN eliminates the FNT and FET entries for this file.

When TEST is called, a portion of the file is read and stored into the IBUFF array, and the variables RTCODE, NWDS, and LEVEL are set. These variables are defined as follows:

```
RTCODE = 1 if an EOR mark is read
RTCODE = 2 if an EOF mark is read
RTCODE = 3 if an EOI mark is read
```

Otherwise RTCODE = 0. Also

```
NWDS = the number of words that are read into IBUFF
LEVEL = the level number of the record.
```

A maximum of 2048 words are read. If an EOR, EOF, or EOI mark is encountered then reading terminates. In this case NWDS is the number of words read that precede the mark.

The TEST routine operates as follows. The first time that TEST is called the first NWDS words of the file are read. The next time that TEST is called the next NWDS words of the file are read, etc. If an EOR or EOF mark is read (which to the author's knowledge should never occur), then TEST can be called again to read the information that follows the mark. However, when an EOI mark is read, then this mark terminates the file and no further reading is possible. Then the PFRETRN routine must be used.

If the words of a pru are numbered from 0 through 63, then an EOR, EOF, and EOI mark affects the count of the prus as follows. If n - 1 prus have been read and the first w words ($0 \leq w < 63$) of the n-th pru are followed by a mark, then the mark is considered to be located at word w of the pru and to terminate the pru. In this case we have what is called a <u>short pru</u>. Any information that follows the mark is considered to begin at word 0 of pru n + 1.

Note. Any program that employs the GETFILE routine must be a system overlay that is invoked by the console operator. This limits access to the PFD and PFC.

RB Chains

An RB chain consists of one or more word pairs. We review briefly the structure of these word pairs. For additional information see pages 4-35 through 4-37, II-1-110, and II-3-7 of reference (1).

48	39	36	24	12	0
link	ord	k	RB_0	RB_1	RB_2
RB_3	RB_4		RB_5	RB_6	RB_7

Note the above representation of a word pair. Bytes 2-4 of the first word of the word pair and bytes 0-4 of the second word of the word pair are called the $RB_0,\ldots,RB_7$ bytes respectively. Byte 0 of the first word contains the link to the next word pair in the RB chain, if there is one. Otherwise, if the word pair is the last word pair of the chain then byte 0 contains the value 0.

If the word pair references one or more RB's, then all the RB's referenced by the word pair must have the same RBR ordinal. If ord is the RBR ordinal then the first 9 bits of byte 1 of the first word contain this ordinal and the remaining 3 bits of byte 1 contain the index k of the first byte RB_k that references an RB. Byte RB_k contains the number n ($1 \le n \le 3232$) of the RB that is being referenced, and $RB_{k+1},\ldots,RB_7$ contain the numbers of the remaining RB's that are being referenced by the word pair. The first byte RB_m ($m > k$) that contains the number 0, if there is one, terminates this list. (Then byte RB_{m-1} contains the number of the last RB that is referenced by the word pair.)

It is possible that the index k refers to a byte RB_k that contains the number 0. If this occurs then the word pair references no RB's and an RBR ordinal is not needed. However, the only case (to the author's knowledge) when an ordinal is not given is when the first 9 bits of byte 1 contain the value 777 (octal) and the index k is set to 7. Then, of course, RB_7 contains the value 0.

SUBROUTINE RBSET(CHSIZE,NUM,DSKRBS,CHAIN)

The first prus of the PFD and PFC contain only the RB chains for the PFD and PFC. The RBSET routine examines the RB chain for the file under consideration (the PFD or PFC). All variables used in this routine are integer variables. If an error is detected which requires that the job be terminated prematurely, then the variable WD is set to 0 in order to inform the main program of this fact. (This variable, as well as many of the other variables employed by the routine, are declared in COMMON statements.) After the SETUP routine has finished, the main program first calls the RBSET routine by the statements

```
FILE = 0
CALL GETFILE
WD = 400000B
CALL RBSET(PFDSIZE,PFDMRK,PFDRBS,PFDCHN(1))
```

for checking the RB chain of the PFD. After this checking has been completed, the main program calls PFRETRN to close the PFD file. If no fatal errors have been detected then the RBSET routine is recalled by the statements

```
FILE = 1
CALL GETFILE
WD = 200000B
CALL RBSET(PFCSIZE,PFCMRK,PFCRBS,PFCCHN(1))
```

for examining the RB chain of the PFC. After this analysis is completed, unless a fatal error has been detected, the PFC file is left open.

The RBSET routine operates as follows. In the statements preceding statement 100, RBSET first checks if there are any EOR, EOF, or EOI marks preceding the RB chain. If there is an EOI mark or if there are 50 or more EOR and/or EOF marks, then this is a fatal error and the system proceeds to the Error Processor section of RBSET. Otherwise, if EOR and/or EOF marks exist, but there are less than 50, then an error message is printed but analysis is not terminated. This is the only error that is detected by RBSET that is not judged to be serious enough to warrant a premature abort. In this case, however, disk addresses cannot be given for the data in the file. Upon the termination of the checking for the marks, if an EOI mark has not been detected then the argument NUM has as its value the number of EOR and/or EOF marks preceding the RB chain of the file. From the above CALL statements we note that this defines the variables PFDMRK and PFCMRK as follows:

```
PFDMRK = the number of EOR and/or EOF marks that precede the RB chain
         for the PFD
PFCMRK = the number of EOR and/or EOF marks that precede the RB chain
         for the PFC
```

These variables are used throughout the program for determining if disk addresses can or cannot be given. Disk addresses for information in the PFD can only be given if PFDMRK = 0, and disk addresses for information in the PFC can only be given if PFCMRK = 0.

If the only marks found to precede the RB chain are NUM EOR and/or EOF marks where $0 \leq NUM < 50$, then RBSET begins its analysis of the RB chain at statement 100. From the above CALL statements we note that

CHSIZE = the maximum number of prus that the RB chain for the file can take.[1]

Thus the variable MAXWPR defined in statement 100 is the maximum number of word pairs that the RB chain can have. Each time that a pass is made through the DO loop that begins here a different word pair of the chain is analyzed. The word pairs are taken in sequence. If more than MAXWPR word pairs are found then we have a fatal error. Upon exiting from the DO loop (at the statement following statement 131) the system will proceed to statement 242 of the Error Processor section of RBSET.

Analysis of a word pair is as follows. First the index I of the first word of the word pair in IBUFF is checked. If $I \geq NWDS$ (NWDS is the number of words of the RB chain that were read by TEST), then an EOR, EOF, or EOI mark appears in the middle of the RB chain and the system is instructed to proceed to statement 240 of the Error Processor section. Otherwise, if $I < NWDS$ then the link, the RBR ordinal ORD of the RB's referenced by the word pair, and the index IND of the first byte that references an RB are obtained. If $ORD > NUMTBL$ (NUMTBL is the largest RBR ordinal that can be used) then we have a fatal error and the system is instructed to proceed to statement 200 of the Error Processor section. Otherwise, the (M,N) coordinates of the ordinal are obtained. If the RBR ordinal is not listed as being used, then again we have an error and again the system is told to go to statement 200. Otherwise, it is checked if the index IND is 0 or 3. If $IND \neq 0,3$ then it is <u>not</u> assumed that there is an error. To the author's knowledge this will not occur. However, to be on the safe side, a message is printed and then analysis continues.

The inner DO loop beginning the third statement after statement 121 is for checking the bytes that reference the RB's. However, this loop processes the bytes of only one word of the word pair. Thus the loop must be entered twice if $IND < 3$. If $IND \geq 3$ then K is assigned the value 2 and the inner DO loop is entered. Upon exiting from the loop, the bytes of the second word of the word pair have been checked and we are finished with the RB bytes. However, if $IND < 3$ then K is first assigned the value 1, the value IND is increased by 5, and the inner DO loop is entered. Upon exiting from the loop the bytes of the first word of the word pair will have been checked. In order to check the bytes of the second word K is reset to 2, IND is reset to 3, and the inner DO loop is reentered.

The operation of the inner DO loop is quite simple. Each time that a pass is made through the loop a different byte is checked. The bytes are taken in sequence. In a pass through the loop, first the number of the RB that is being

---

[1] The variables PFDSIZE and PFCSIZE, which are defined on page A-1, are used here only.

referenced by the byte is obtained. This number is denoted by the variable RB. If RB = 0 then the RB's referenced by the word pair have all been previously checked (if there were any) and we are done. Otherwise, if RB > 3232 then we have a fatal error and the system is instructed to proceed to statement 202 of the Error Processor section. However, if this error does not occur then RBSET next checks if the 20 bit field for the RB in the RBRTBL is 0. If not then we have a conflict and the system is instructed to go to statement 210 of the Error Processor. Otherwise, if no conflict is found, then the RB reference is considered to be correct. In this case, the value of the variable DSKRBS is increased by 1 (DSKRBS is the number of RB's in the RB chain that have been examined) and the 20 bit field for the RB in the RBRTBL is modified to contain the number of the word that contains this byte. By the way in which the variable WD is defined, being given an offset value of 400000B or 200000B initially, the coding will also automatically set the PFD or PFC bit, whichever is appropriate. Then the pass through the inner loop is complete.

After the RB bytes have been examined, if no errors have been detected then RBSET will arrive at statement 130. Here it is checked if the word pair is the final word pair of the chain. This completes the analysis of the word pair. Upon the termination of the checking of the word pairs of the RB chain, no matter whether an error has been detected or not, we note from the CALL statements at the beginning of this section that DSKRBS defines the variables PFDRBS and PFCRBS as follows:

PFDRBS = the number of RB's in the RB chain for the PFD which were examined and for which no errors were detected
PFCRBS = the number of RB's in the RB chain for the PFC which were examined and for which no errors were detected

These variables are also used throughout the program for determining if disk addresses can or cannot be given.

After the RB chain has been examined, if no fatal errors have been detected then at statement 140 the RB chain is copied into the array CHAIN. From the CALL statements at the beginning of this section it follows that

PFDCHN contains the RB chain for the PFD and
PFCCHN contains the RB chain for the PFC.

The RB chains are saved since they may be needed later for computing disk addresses. Next RBSET updates the variables WD and I to refer to the beginning word of the next pru. Also it checks if the current pru contains an EOI mark. If the pru does contain an EOI mark, then the mark necessarily must follow the RB chain but we still have a fatal error. Then the system is directed to proceed to statement 260 of the Error Processor. If no such EOI mark exists, and if no EOR and/or EOF marks precede or immediately follow the RB chain in the same pru, then the RBSET routine is finished. Otherwise, a dump of the RB chain is written on tape 1. If an EOR or EOF mark follows the RB chain then the TEST routine is recalled and I is reset. Then RBSET terminates.

Note All dumps that are to be printed are first written on tape 1. Then at the end of the job, at statement 310 in the main program, tape 1 is printed and the main program terminates.

The Error Processor Section

Whenever a fatal error is detected this section is entered. Here error information is printed and (whenever possible) a dump is written on tape 1. Also the variable WD is set to 0, and then RBSET terminates. After RBSET has finished the main program closes the file, prints tapes 10 and 1, and then aborts. (Tape 10 has not yet been discussed. In this case, however, only tape 1 need be considered. Tape 10 will contain no information.)

The coding in this section is fairly straightforward, requiring only the use of the ADDRSS, DSKADD, and PRTPRU routines. The routine ADDRSS is used only by RBSET. Its brief definition

```
      SUBROUTINE ADDRSS(M,I,RB,PRU,WORD)
      INTEGER RB,PRU,WORD

C     M IS A 20 BIT POSITIVE INTEGER (RIGHT ADJUSTED) THAT REFERENCES
C     A WORD IN THE RB CHAIN FOR THE RBTC OR PFD. THIS ROUTINE DETER-
C     MINES THE ADDRESS (RB,PRU,WORD) OF THE WORD. I IS SET TO 3 IF
C     THE WORD IS IN THE RB CHAIN FOR THE PFC, AND I IS SET TO 4 IF
C     THE WORD IS IN THE RB CHAIN FOR THE PFD.

      I=SHIFT(M,-17)+3
      WORD=M.AND.77B
      PRU=SHIFT(M,-6).AND.1777B
      RB=PRU/56
      PRU=PRU-RB*56
      RB=RB+1
      END
```

requires no comment. The DSKADD routine is used for obtaining disk addresses, and the PRTPRU routine writes a pru on tape 1. These two routines are described in detail in the next two sections.

As an example of the operation of the Error Processor, consider the case where there is a conflict. At statement 210 first the address (NUMRB,PRU,W) of the word that references the RB in conflict is obtained. Also the variable INFO is defined to reference a word that contains the following information:

```
bits  0-19 contain the 20 bit field of the RB in the RBRTBL
           that is in conflict
bits 20-59 contain zeros
```

Then a statement is printed which asserts that a conflict exists, and RBSET checks the flaw bit of the 20 bit field in INFO. If the bit is 0 then the RB in

conflict is not flawed and the system is instructed to go to statement 220. Otherwise, the system is instructed to print the fact that the RB is flawed and to print the address of the word that references this RB. Now if the RB is unflawed, then at statement 220 the address (NUMRB1,PRU1,W1) of the previous word that referenced the RB is obtained. If the previous reference was made in the same file, then the system is instructed to proceed to statement 230. Otherwise, the first reference must have been from the PFD and RBSET must now be examining the PFC. In this case, the system first prints the addresses of the words that reference the RB in conflict. Then it checks if disk addresses can be obtained for the RB's in the PFD. If this is possible, then the disk address for RB NUMRB1 is found and printed. At statement 208 the system next checks if disk addresses can be obtained for RB's in the PFC. If this is possible,then the disk address for RB NUMRB is found and printed. Thereupon the system arrives at statement 300.

Whenever statement 300 is reached, all the error information has been printed and the only tasks left are to set WD to 0 and to write a dump on tape 1. In the coding beginning at statement 310 a dump is written. Unless an EOI mark or several EOR and/or EOF marks are encountered in the middle of the chain, the dump will contain the entire RB chain and several succeeding prus. After the writing is finished RBSET terminates.

Disk Addresses

Assume that there are no EOR, EOF, or EOI marks at the beginning of the file under consideration (the PFD or PFC). Consider the n-th RB of the file. Then in order to find where the RB is stored on disk, first find the n-th RB that is referenced by the RB chain for the file. If the word pair that references this RB contains the RBR ordinal ORD, and if m is the number of the RB being referenced (this number is stored in the byte that references the RB), then the pair of numbers (ORD,m) tell us where on disk the n-th RB of the file is stored. We shall refer to (ORD,m) as the disk address of RB n.

Example Since it is assumed that the RB chain for the PFD begins in the first RB of the PFD, in order to find where on disk this chain is located it is necessary to find the first RB that is referenced by the RB chain. The first word pair of the RB chain will normally reference an RB. If this occurs then the RBR ordinal ORD in this word pair and the number m stored in the first byte that references an RB tell us where on disk the RB chain is located. We then know that the RB chain (which resides in the first RB of the PFD) is stored on disk in RB m of RBR ordinal ORD. In this case RB 1 has the disk address (ORD,m).

SUBROUTINE DSKADD(NUMRB,CHAIN,ORD,RB)

Assume now that we wish to obtain the RBR ordinal (ORD) and the disk RB number (RB) for RB NUMRB of the PFD or PFC. Let the CHAIN array contain the RB chain for the PFD or PFC. Assume that no EOR, EOF, or EOI marks precede the RB chain, and that NUMRB is less than or equal to the number of RB's in the RB chain that have been examined and found to be correct. Then the DSKADD routine finds the disk address (ORD,RB) for RB NUMRB. The procedure adopted for examining the word pairs of the RB chain is similar to that used by the RBSET routine. The coding from statement 100 to the end of the routine comprises a loop. Each time that a pass is made through the loop a different word pair of the chain is examined. The word pairs are taken in sequence.

Analysis of a word pair is as follows. First the RBR ordinal ORD and the index IND of the first byte that references an RB are obtained. Then the DO loop beginning at statement 121 is entered. This DO loop is for counting the bytes that reference RB's. However, this loop processes the bytes of only one word of the word pair. Thus the loop must be entered twice if IND $\leq$ 3. If IND $\geq$ 3 then K is assigned the value 2 and the DO loop is entered. Upon exiting from the loop, the bytes in the second word of the word pair have been counted and we are finished with this word pair. However, if IND $<$ 3 then K is first assigned the value 1, the value of IND is increased by 5, and the DO loop is entered. Upon exiting from the loop the bytes of the first word of the word pair have been counted. In order to count the bytes of the second word K is reset to 2, IND is reset to 3, and the DO loop is reentered.

The operation of the DO loop is quite simple. Each time that a pass is made through the loop a different byte is examined. The bytes are taken in sequence.

In a pass through the loop, first the number of the RB that is stored in the byte is obtained. This number is denoted by the variable RB. If RB = 0 then the RB's referenced by the word pair have all been previously counted (if there were any) and we are finished with this word pair. Otherwise, if RB $\neq$ 0 then the value of the variable NUM is increased by 1 (NUM is the total number of RB references that have been encountered). If NUM = NUMRB then we are done. The current values for ORD and RB are the values that were needed and the routine terminates. Otherwise, if NUM $\neq$ NUMRB then NUM is still less than NUMRB and this pass through the DO loop is complete.

SUBROUTINE PRTPRU(I,PRU,MEM,NUMWDS,ICODE,K,IND)

The PRTPRU routine is used in writing dumps. The array MEM contains a pru to be printed. PRTPRU has the task of writing this pru on tape 1. The arguments I, PRU, and NUMWDS are assumed to have the values:

I = the index of the beginning word of the pru in MEM
PRU = the number of the pru
NUMWDS = the number of words in MEM

If $I+63 \leq NUMWDS$ then MEM(I),MEM(I+1),...,MEM(I+63) are the words of the pru. The argument ICODE is ignored, thereby permitting it to be set arbitrarily. Otherwise, if $I+63 > NUMWDS$ then the pru is a short pru, being terminated by an EOR, EOF, or EOI mark. The argument ICODE informs PRTPRU which type of mark terminates the pru. ICODE has the format:

ICODE = 1 if an EOR mark terminates the pru
ICODE = 2 if an EOF mark terminates the pru
ICODE = 3 if an EOI mark terminates the pru

If the pru is short and $I \leq NUMWDS$, then MEM(I),MEM(I+1),...,MEM(NUMWDS) are the words of the pru. However, if $I > NUMWDS$ then it is assumed that the pru to be written contains no words.

Note Because of the way in which the IBUFF array is used, if MEM begins at the m-th word of IBUFF, then it is assumed that I = 2049-m ; i.e., it is assumed that I refers to one of the first 2048 words of IBUFF.

PRTPRU operates as follows. First it is checked if $I+63 \leq NUMWDS$. If the pru is a short pru then IND is set to 1, MAX is set to NUMWDS, and the system is instructed to go to statement 20. Otherwise, if the pru contains 64 words, then IND is set to 0 and (in the loop beginning at statement 10) MAX is defined to be the index of the last word of the pru that contains nonzero information. (In the case that the pru contains only zeros, then MAX = I-1.) The purpose of IND, which is an argument of PRTPRU, is to inform the user of PRTPRU if the pru is short or not.

In the coding from statement 20 to statement 30, a title line is written for the pru. This line contains the information NUMRB,NUMPRU where

NUMRB = the number of the RB that contains the pru and
NUMPRU = the number of the pru in this RB.

If the argument K of PRTPRU has been assigned a nonzero value, then NUMRB,NUMPRU is the only information that is given in the line. Otherwise, if K = 0 then the disk address of RB NUMRB is also written.

At statement 30 it is first checked if the pru contains any information. If not, then at statement 40 the type of the mark that terminates the pru is written[1]

[1] It should be noted that this is one of the major spots in the program where the MARK array is used. This array is defined at the beginning of the main program.

and the routine ends. If, however, the pru contains information, then it is next checked if I > MAX. This can occur only if the pru consists of 64 words, all of which contain only zeros. In this case, at statment 50 a message is written and the routine terminates. Otherwise, if the pru is short or contains nonzero information, then the statement

```
WRITE(1,32) (IADD1(WORD),MEM(J),IBLANK(MEM(J)),J=I,MAX)
```

writes the words on tape 1. After this is done, if the pru is short then at statement 40 the type of mark that terminates the pru is written. Then the routine ends.

The above WRITE statement operates as follows. First the function IADD1 is called. Its brief definition

```
INTEGER FUNCTION IADD1(N)
IADD1=N
N=N+1
RETURN
END
```

may appear a bit strange. If the value of the variable N is n, then after IADD1(N) has been computed, IADD1(N) = n and N = n + 1. Thus the n-th time that IADD1(WORD) is called by the WRITE statement, IADD1(WORD) = n-1 and WORD = n. Hence, IADD1(WORD) will have as its value the current number of the word in the pru that is to be written next.

The word to be written next, namely MEM(J), is to be written in both Ø20 and A10 formats. For the latter format the function IBLANK defined by

```
      INTEGER FUNCTION IBLANK(N)
      IBLANK=N
      DO 10 I=1,10
      IBLANK=SHIFT(IBLANK,6)
      IF ((IBLANK.AND.77B).EQ.0) IBLANK=IBLANK.OR.55B
10    CONTINUE
      RETURN
      END
```

is needed. Here N references a word. IBLANK(N) is a modification of N where the colons (having display code 00B) have been replaced with blanks (having display code 55B). The modified word IBLANK(MEM(J)) is the word that is written in the A10 format.

PFC Entries

As was noted earlier, the first prus of the PFC contain the RB chain for the PFC. Consider now the remaining prus, which contain the PFC entries for the permanent files. Word 0 of each pru serves the purpose of indicating the status of the pru. The word contains the information:

Bit 1 = 1 if the pru is being used and 0 otherwise.
Bit 4 = 1 if the pru contains an entry for a file in the I/O queue and 0 otherwise.
The remaining bits are 0.

If bit 1 of the word is 1, then words 1-62 of the pru contain a PFC entry (or a portion of a PFC entry) for a permanent file. (Word 63 of the pru is reserved.)

Only one PFC entry can reside in a pru. If an entry occupies two or more prus, then the prus being occupied are contiguous, following one another in sequence. The initial pru of an entry has the format given on pages II-3-6, II-3-8, and II-3-9 of reference (1). Of particular interest are the number L and K stored in byte 3 of word 11 and byte 2 of word 12 respectively.[1] Then

K = the number of words following word 12 that precede the RB chain
L = the total number of words in the entry.

If n is the number of words in the RB chain for the entry, then L can be defined more precisely by

$$L = K + 12 + n.$$

Thus if $L = K + 12$ then the entry has no RB chain. (This can occur.) To the best of the author's knowledge, $11 \leq K \leq 48$ must always be satisfied. The restriction $K \leq 48$ ensures that the RB chain must begin in the first pru of the entry. An RB chain word pair can never begin in one pru and end in another. Let $m = 0$ if K is odd and $m = 1$ if K is even. Then words $K + 13$ through $61 + m$ of the first pru of the entry provide enough space for $\frac{1}{2}(49 + m - K)$ word pairs of an RB chain for the entry. If additional storage is needed, then one or more continuation prus may be used. Each continuation pru provides storage for 31 additional word pairs. The word pairs are stored in words 1-62 of a continuation pru.

[1] These numbers are in decimal.

SUBROUTINE RBTCHK

RBTCHK is the general routine for analyzing PFC entries. After the RB chains for the PFD and PFC have been examined by RBSET, if no fatal errors have been detected then the main program sets IERR = PFDMRK + PFCMRK, modifies WD to be the number of words in the PFC that have been processed, and then calls RBTCHK. Currently the variable I is the index of the word in IBUFF that is word 0 of the first pru in the PFC that follows the RB chain for the PFC. Thus if IBUFF(I) contains word 0 of pru m then WD = (m-1) * 64. This arrangement is provided by RBSET in the coding beginning the second statement after statement 140.

Very little error processing is done by RBTCHK. Whenever an error is detected a 5 word message is written on tape 2, the value of IERR is increased by 1 (if the error is not a conflict), and the value of MCOUNT is decreased by 1. Then if MCOUNT = 0 the routine terminates. Otherwise, RBTCHK continues its analysis of the PFC. The variable IZERO is used in writing messages on tape 2 and IERR has the definition:

IERR = the number of errors found that are not conflicts.

Messages may be issued for situations that occur that are not errors. The first word of a 5 word message indicates the type of situation and the second word contains the number of the word in the PFC at which the situation occurred or was detected. Only MCOUNT messages will be written on tape 2. Then analysis of the PFC by RBTCHK is terminated. The variable MCOUNT is defined in the main program. Its value cannot exceed 4000. This restriction is not needed here, but is required in later routines. When RBTCHK terminates, tape 2 will reference (among other things) all errors that were found by RBTCHK. It will then be the task of the later routines to examine each of the 5 word messages, and to report on them to the user.

The layout of RBTCHK can briefly be summarized as follows. All the statements from statement 1 to the end of the routine comprise a single loop. One pru is examined each pass through the loop. The prus are taken in sequence until

(1) MCOUNT messages have been written on tape 2 or
(2) the prus of the PFC have all been examined.

Then the routine terminates. (None of the errors detected by RBTCHK are considered to be fatal, requiring a premature abort of the job.) The analysis of a pru normally begins at statement 10. However, if the variables NUM, IPRU, and WDPRU have not been set then analysis begins at statement 1.

All variables used by RBTCHK are integer variables. I,WD,IPRU,WDPRU,NUM have the definitions:

I = the index of the word in IBUFF to be examined next.
WD = the number of words in the PFC that have already been examined.
IPRU = the index of word 0 of the current pru in IBUFF.
WDPRU = the number of words in the PFC that precede the current pru.
NUM = the total number of words in the 64 word prus in IBUFF.

Thus at the beginning of the analysis of a pru, IPRU = I and WDPRU = WD. The variable L is also important. At the beginning of a pru L is normally 0. If $L \neq 0$ then a PFC entry that occupies more than a pru is being examined. In this case, the pru under consideration is a continuation pru for the entry and L is the number of words in the RB chain for the entry that have not yet been checked.

The operation of RBTCHK is as follows. Statement 1 is encountered when RBTCHK first begins and whenever a new portion of the PFC has been read and stored in IBUFF by the TEST routine. I and WD are assumed here to have already been defined. RBTCHK first checks if NWDS = 0. If this is the case, then the pru under consideration contains no words and the system is instructed to go to statement 11 (see the next paragraph). Otherwise, if NWDS > 0 then NUM, IPRU, and WDPRU are defined and the system arrives at statement 10.

The normal flow of control begins at statement 10. Here it is checked if the pru under consideration contains 64 words. If it does then the system proceeds to statement 20. Otherwise, RBTCHK next checks if NWDS = 2048. If this is the case, then all the words in IBUFF have been examined and the system is instructed to go to statement 3. Otherwise, if $NWDS \neq 2048$ then the pru is a short pru. In the coding beginning at statement 11 a message is written on tape 2, L is reset to 0, IERR is increased by 1 if the pru is not terminated by an EOI mark, and MCOUNT is decreased by 1. If MCOUNT = 0 or an EOI mark terminates the pru, then the routine ends. Otherwise, WD is increased by 64 and the system arrives at statement 3. Then a new portion of the PFC is read and stored in IBUFF by the TEST routine. Also I is reset to 1 and the system returns to statement 1.

The block of coding beginning at statement 20 and continuing to the end of the routine is devoted entirely to the treatment of 64 word prus. If the pru under consideration is a 64 word pru, then it arrives at statement 20 of this block after leaving statement 10. After analysis of the pru has been completed, the system will exit from the block in the coding beginning at statement 500 or 501. Here IPRU,WDPRU,I,WD will be reset to prepare the system for the next pru to be examined. Also L will be reset to 0 if this is necessary. Then the system will return to statement 10.

Tape 2 messages are handled in a systematic manner in the block. If the message concerns a situation that is not an error, or if it refers to an error whose effects are local, then in the coding the message is processed where the situation is detected. After the processing is finished, if $MCOUNT \neq 0$ then analysis continues. Otherwise, if the message refers to an error, the existence of which leaves in doubt either the validity of or the handling of the remaining data in the 64 word pru, then analysis of the pru is terminated. In the section of coding beginning at statement 400 the error is processed. After this is done, if $MCOUNT \neq 0$ then at statement 500 or 501 preparation is made for the analysis of the next pru.

The section of coding beginning at statement 20 directs the operation of the block. If $L \neq 0$ then the pru under consideration is a continuation pru for a PFC entry and the system is instructed to go to statement 300. Otherwise, if $L = 0$ then it is first checked if word 0 of the pru (the word that indicates the status of the pru) is 0. If this is the case, then the pru is not being used and the system is instructed to go to statement 501. However, if word 0 of the pru is not 0, then it is next checked if word 0 has the proper format. If not, then the system is instructed to go to statement 400. Otherwise, if word 0 has the proper format, then the pru is the beginning pru of a PFC entry. I and WD are now updated and the system arrives at statement 100.

The coding beginning at statement 100 processes the first words of a PFC entry. It is first checked if word 1 has the proper format. If not, then the system is instructed to go to statement 400. Otherwise, L and K are defined to be the numbers stored in byte 3 of word 11 and byte 2 of word 12 respectively. (See the discussion on page 19.) If $L = K + 12$ then the entry has no RB chain. In this case, analysis of the pru is complete and the system is instructed to go to statement 500. Otherwise, if $L \neq K + 12$ then it is next checked if L-K is even, $L \geq K + 14$, and $11 \leq K \leq 48$. If any of these conditions is violated, then we have an error and the system is instructed to go to statement 402. However, if all the conditions are satisfied then the preliminary analysis of the entry is finished. A message is written on tape 2 if the entry occupies more than a pru, and L,I,WD,MAX are assigned the values:

L = the number of words in the RB chain.
I = the index of the word in IBUFF that is the first word of the RB chain.
WD = the number of words in the PFC that precede the RB chain.
MAX = the maximum number of word pairs of an RB chain that the pru can contain.

Also the ENTRY array is defined and the system arrives at statement 200. Henceforth, for the remainder of the analysis of the entry, L will be the number of words in the RB chain for the entry that have not been examined.

In general, no matter whether RBTCHK arrives at statement 200 via statement 100 or statement 300, the situation will be the same. If at statement 20 the pru is found to be a continuation pru for a PFC entry, then the following sequence of events occur. First the system proceeds to the section beginning at statement 300. Here word 1 of the pru is checked. If the word contains a label for a PFC entry, then we have an error and the system is instructed to go to statement 400. Otherwise, there is no further preliminary processing for the system to do. It must be assumed that only word pairs of the RB chain for the entry are stored in the pru. Then I, WD, MAX, and ENTRY are updated and the system proceeds to statement 200.

The DO loop beginning at statement 200 examines the word pairs of the RB chain that are in the pru. The structure of this loop requires no comment, being almost the same as the structure of the DO loop in RBSET that begins the first statement after statement 100. The following three details, however, are of interest and should be kept in mind:

(1) If an RBR ordinal ORD is encountered that is not the ordinal for a permanent file device that is on, then we may or may not have an error. Since certain temporary files (normally I/O files) can reside on nonpermanent devices, there is often the possibility that ORD may be the RBR ordinal for a nonpermanent file device.

(2) Conflicts are the only errors found in this section that are not considered to be serious enough to terminate the processing of the pru. If a conflict is found in the inner DO loop beginning at statement 221, then the error is processed in the inner DO loop. After this is done, if MCOUNT $\neq 0$ then analysis continues of the remaining bytes of the word.

(3) The array ENTRY, whose purpose is to hold information that may be inserted in the RBRTBL, is defined much earlier in RBTCHK than in RBSET. This can be done since the 20 bit fields (in the RBRTBL) for the RB's referenced in the RB chain for the entry now contain pru numbers rather than word numbers.

In operation, RBTCHK analyzes the PFC extremely quickly. The time that the routine takes can be expected to increase linearly with the number of permanent files. After the routine terminates the PFC is closed by PFRETRN, a five word message consisting entirely of zeros is written on tape 2, and tape 2 is rewound. This finishes the first half of the program (the analysis portion). Now the messages on tape 2 must be examined and reported on.

Note Tape 2 should have been treated as an unformatted binary file. This would have made virtually no difference in the timing, but the coding would have been a bit cleaner. As it stands, tape 2 is a file consisting of five word messages, each of which has the format 5Ø20.

SUBROUTINE INFOCON(MCOUNT)

After RBTCHK terminates, the main program redefines MCOUNT to have the value:

MCOUNT = the number of messages that were written on tape 2
by RBTCHK

Also messages are issued to the dayfile and output file informing the user of the number of errors found that are not conflicts. After this is done the INFOCON routine is called. This routine reads tape 2, collecting together all the conflicts found by RBTCHK that do not involve flawed RB's or RB's in the RB chains for the PFD and PFC. While doing this the following variables are defined:

MPFD = the number of conflicts with the RB chain for the PFD
MPFC = the number of conflicts with the RB chain for the PFC
MFLAWS = the number of conflicts involving flawed RB's
MAX = the number of conflicts that do not involve flawed RB's
or RB's in the RB chains for the PFD and PFC

After the conflicts have been collected, in the coding following statement 113 INFOCON rewinds tape 2 and issues messages to the dayfile and output file reporting the information MPFD,MPFC,MFLAWS,MAX. Then INFOCON terminates.

The conflicts not involving flawed RB's or RB's in the RB chains for the PFD and PFC are collected together and stored in the array CON. CON contains 24000 words. In order to conserve storage, CON is permitted to occupy the storage space that RBRTBL occupies. This can be done since the RBRTBL array is no longer needed. CON has two sections. The first section consists of the first 20000 words of CON. The second section, which is the subarray CONPRU, consists of the last 4000 words of CON.

The first section of CON contains the conflict entries. The first two words of an entry contain the RBR ordinal (ORD) and the disk RB number (RB) of an RB that is in conflict. The format of the entry is ORD,RB,n,WD_1,...,WD_n. Thus the entry contains n + 3 words. The word WD_1 contains the pru number (shifted to the left 6 bits) that was stored in the RBRTBL array for this RB. The n - 1 words WD_2,...,WD_n contain the word numbers of the words in the PFC that subsequently tried to reference this RB (when it was already known to be in use). An entry in the first section of CON must contain at least five words. Each entry refers to a different RB in conflict. The entries are stored sequentially, being ordered by the values of ORD and RB. If MAX is the number of conflicts, then the maximum amount of storage required for the conflicts in the first section of CON is 5 * MAX. Since 20000 words are being allotted for this section, the section can contain at most 4000 entries. This limits the number MCOUNT of messages that can be issued by RBTCHK to 4000.

The second section of CON, namely the subarray CONPRU, contains the numbers of the prus in the PFC that first referenced the RB's in conflict. These numbers are listed in increasing order.

The coding of INFOCON is fairly self-contained, requiring only the aid of the SHIFT and ASHIFT routines. ASHIFT is a routine for shifting data in an array. If A is an array of dimension n then

```
CALL ASHIFT(A,n,k)
```

shifts the data in A k words to the right. The coding

```
      SUBROUTINE ASHIFT(A,N,K)
      INTEGER A(N)
C     HERE IT IS ASSUMED THAT K IS A POSITIVE INTEGER EQUAL TO OR LESS
C     THAN N. THE ROUTINE SHIFTS THE DATA IN A(1),...,A(N-K) K WORDS TO
C     THE RIGHT.

      IF (K.EQ.N) RETURN
      I=N
      M=N-K
      DO 10 J=1,M
      A(I)=A(I-K)
10    I=I-1
      RETURN
      END
```

which defines the routine requires no comment.

INFOCON collects the conflicts as follows. First the variables MAX,MPFD,MPFC, MFLAWS are set to 0 and the storage area for CON is cleared. Then the DO loop following statement 1 is entered. This DO loop reads the MCOUNT messages written on tape 2 by RBTCHK. Each time that a pass is made through the loop a different message is read. The messages are taken in sequence. (MCOUNT must always be greater than 0. If MCOUNT $\leq$ 4000 then tape 2 contains a message concerning the EOI mark for the PFC.)

Consider now the i-th pass through the loop. First the i-th message on tape 2 is read and it is checked if the message refers to a conflict. If not, then we are finished with the message and the system is instructed to go to statement 113 (which terminates this pass through the loop). Otherwise, if the message refers to a conflict then (ORD,RB) is the disk address of the RB in conflict, WD is the number of a word in the PFC that references this RB, and INFO is a word containing the information:

bits 0-19 contain the 20 bit field of the RB in the RBRTBL that is in conflict
bits 20-59 contain zeros

Letting W = SHIFT(INFO,-16) + 1 then

$$W = \begin{cases} 5 & \text{if the conflict involves a flawed RB} \\ 3 & \text{if the conflict is with the RB chain for the PFD} \\ 2 & \text{if the conflict is with the RB chain for the PFC} \\ 1 & \text{otherwise} \end{cases}$$

Depending on the value of W, the count (MFLAWS, MPFD, MPFC, or MAX) of the number of conflicts of the type involved is increased by 1. If $W \neq 1$ then we are finished and the system is instructed to go to statement 113. Otherwise, in the coding immediately following statement 14 the number of the pru (INFO) that first referenced the RB in conflict is inserted in CONPRU. After doing this the system arrives at statement 100.

In the section beginning at statement 100, a search is made for a conflict entry in CON that references the RB having disk address (ORD,RB). If none is found, then at statement 102 the 5 word entry

ORD,RB,2,INFO,WD

for the conflict is inserted in the first secion of CON. Otherwise, if an entry ORD,RB,n,WD_1,...,WD_n is found that references the RB in conflict, then WD_1 = INFO and at statement 112 the entry is changed to

ORD,RB,n + 1,WD_1,...,WD_n,WD .

After a new entry is inserted or an existing entry is modified, then the pass through the loop is complete.

Preliminary Considerations concerning the Processing of Errors

After INFOCON terminates, it is necessary to modify the contents of word WD_1 of each conflict entry ORD,RB,n,WD_1,...,WD_n in CON. WD_1 currently contains the number of the pru in the PFC (shifted 6 bits to the left) that first references the RB in conflict. (ORD,RB) is the disk address of the RB in conflict. Word WD_1 must be modified to contain the number of the word in the pru that first references this disk RB. This task requires that the PFC be read a second time by the TEST routine. If the task could be avoided (which it cannot), then the entire program could be restructured so as to avoid a second reading of the PFC.

REPORT is the routine that has the primary responsibility of reporting the results of the analysis done by RBTCHK. If there are errors in a pru in the PFC then messages are written concerning these errors and a brief dump is given. The dump will contain the pru in question, preceded normally by P prus and followed normally by Q prus. A maximum of DLIMIT such dumps will be given by REPORT. The parameters P,Q,DLIMIT are set at the beginning of the coding of the REPORT routine. It is assumed that $P \geq 2$, $Q \geq 2$, and $DLIMIT \geq 0$.[1] The current values used are $P = 2$, $Q = 2$, and $DLIMIT = 150$.

In the environment in which we are operating, where the PFC is read by TEST and stored in IBUFF, a problem now arises in the fact that we can read forwards, from the beginning of the PFC to the end of the PFC, but we cannot reverse the reading process and read backwards, from the end of the PFC to the beginning of the PFC. The existence of a procedure for reversing the direction of the reading process would be welcomed, since in preparing a dump for a pru that contains an error, it is desired that the pru be given along with P preceding prus. However, since no such procedure exists, we do what we consider to be the next best thing under the circumstances. We provide a queue, the array IBUFF1, consisting of the P prus immediately preceding the first pru in IBUFF.

The array IBUFF1 must be of dimension $65 * (P + 1)$ or greater.[2] A pru entry in IBUFF1 consists of 65 words. The first word of an entry contains the number n of words in the pru ($0 \leq n \leq 64$). If $n = 64$ then the remaining 64 words of the entry contain the 64 words of the pru. Otherwise, if the pru is short, then the next n words of the entry contain the words of the pru and the $(n+2)$-nd word of the entry contains the RTCODE[3] for the mark that terminates the pru. The entries are arranged in sequence, the first entry containing the most recent pru to be inserted. Whenever a new entry is to be inserted, the first $P - 1$ entries already in the queue are shifted 65 words to the right by ASHIFT (thereby eliminating the P-th entry) and then the entry is inserted.

---

[1] If P is reset to a value that is greater than the number of prus containing the RB chain for the PFC, then take note of the discussion at the bottom of page 35.

[2] The P prus in IBUFF1 require $65 * P$ words. However, $65 * (P + 1)$ words are needed since an additional pru is often saved. For further details see page 35.

[3] See page 7 .

SUBROUTINE GETPRU(PRU,I)

In order to report the results of the analysis done by RBTCHK, periodically the PFC must be read and the queue IBUFF1 updated. GETPRU is the routine that is called by REPORT to carry out these tasks. When GETPRU is called it is assumed that

PRU = the number of the pru in the PFC that is wanted
PRU0 = the number of the pru in the PFC that is currently the first pru in IBUFF

and that PRU ≥ PRU0. When this routine terminates, IBUFF will contain the pru that is wanted and I will be the index of the word in IBUFF where the pru begins. (If I > NWDS, which can occur only if NWDS ≠ 2048, then I = NWDS + 1 and the pru under consideration will contain no words.) Also PRU0 and IBUFF1 will be updated, if this is necessary.

For convenience, momentarily let L denote the value of PRU0 when GETPRU terminates. If it is assumed that whenever GETPRU is called to find a pru, no later call for GETPRU will be for finding an earlier pru, then IBUFF1 need only be updated if PRU < L + P. Otherwise, if PRU ≥ L + P then the contents of IBUFF1 will be of no interest.

The coding of GETPRU is short, but a bit tricky. Beginning at statement 11, NUM and L are assigned the values:

NUM = the total number of prus in IBUFF [1]
L = the total number of prus in the PFC that have been read

Then it is checked if PRU < L. If this is the case, then there is almost nothing to be done. PRU0 and IBUFF1 need not be modified. In this case, the system is instructed to go to statement 40 where the index I of the pru is set. Then the routine terminates. Otherwise, if PRU ≥ L then PRU0 is reassigned the value L. Also IBUFF1 is updated if PRU < PRU0 + P, and at statement 10 the next portion of the PFC is read by TEST. After this is done, the system again arrives at statement 11. Now the next pass through this loop begins.

The IBUFF1 array is updated as follows. First M is defined to be the number of prus that are to be inserted into IBUFF1. Then the entries in IBUFF1 are shifted M * 65 words to the right, and it is checked if the first pru to be inserted contains 64 words (i.e., it is checked if NWDS = 2048). If not, then I is set to 1, K is set to 2048, and the system proceeds to statement 30. Otherwise, if the first pru to be inserted is a short pru, then at statement 20 K and N are assigned the values

K = the total number of words in the 64 word prus in IBUFF
N = the number of words in the last pru in IBUFF (which is short)

and the short pru is inserted in the first N + 2 words of IBUFF1. After this

[1] The coding for NUM must take the form that it does, for if NWDS = 64 * n where NWDS ≠ 2048, then IBUFF contains n full prus and one 0 word pru.

is done, at statement 22 it is checked if M = 1. If this is the case, then the update of IBUFF1 is complete. Otherwise, I is set to 66, the value of M is reduced by 1, and the system arrives at statement 30.

In the DO loop beginning at statement 30, M 64 word prus are inserted into IBUFF1. Each time that a pass is made through the loop, a different pru is inserted. The prus are taken in sequence, from right to left. At the beginning of a pass, I and K have the values:

I = the index of the word in IBUFF1 where the next word is to be inserted

K = the index of the last word of the pru in IBUFF that is to be inserted

First the value of K is reduced by 64, thereby setting it to be the number of words in IBUFF that precede the pru which is to be inserted. Then J is defined to be the index of the word in IBUFF where the pru begins, and the pru is inserted into IBUFF1. After this is done, at statement 32 the pass through the loop terminates. After the M prus have been inserted, the update of IBUFF1 is complete.

SUBROUTINE REPORT

After INFOCON terminates, the PFC is opened by GETFILE and the REPORT routine is called. REPORT has the responsibility of gathering together and reporting the results of the analysis done by RBTCHK. If a pru is referenced in the CONPRU array and/or there are messages on tape 2 concerning the pru, then a dump is written on tape 1 for the pru. Each dump has a preface. The first line of the preface identifies the pru and contains the number of the dump. The remainder of the preface contains whatever comments and/or complaints that REPORT wishes to make concerning the pru. As noted earlier, the dump contains the pru under consideration, preceded normally by P prus and followed normally by Q prus. Also, a maximum of DLIMIT dumps are issued by REPORT.

REPORT generates three files. The first file, tape 1, contains the dumps and their prefaces. The second file, tape 10, contains copies of all the prefaces. This file, in effect, summarizes the information found on tape 1. The third file, tape 3, provides background information concerning the files that contain conflicts with other files (or with themselves). Included in this information is the name of the file, its cycle number, ID, and creation date.

All variables used by REPORT are integer variables. T and PRUO have the values:

T = the number of prus referenced in CONPRU that have been reported on + 1

PRUO = the number of the pru in the PFC that is currently the first pru in IBUFF

The variable CARRY assists in the handling of long PFC entries (i.e., entries that occupy two or more prus). Normally CARRY = 0. However, CARRY may have the value 1 when either the previous pru reported on contained a portion of a long entry, or the current pru being examined contains a portion of a long entry.

Since the entire PFC may have to be read a second time, REPORT must assume that the PFC is the same as when it was read by the RBTCHK routine. This requires that no new programs be read into the input queue and that the program be permitted to run to completion without interruption.[1] Periodically REPORT checks if the PFC is the same. If not, then the variable SIG becomes nonzero. Whenever SIG $\neq$ 0 then either the PFC has been modified, there is a bug in the program, or the system has been so badly damaged that the program can no longer run properly. In any case, the job must then be aborted.

The operation of REPORT can be described as follows. All the statements from statement 5 to the statement preceding statement 300 form a loop. Each time that a pass is made through the loop a different pru is reported on. At the beginning of a pass DMPNUM has the value:

[1] RBTCHK requires that no programs in the output queue be printed.

DMPNUM = the number of passes that have previously been made through the loop[1]

Each pass through the loop begins at statement 5 or 10, depending on whether a new message from tape 2 must be read or not. At statement 10 the variables MSGE,WD,INFO,RB,ORD contain the contents of the tape 2 message to be considered next. It is first checked if MSGE = 0. If this is the case then the message is the terminal message that was written on tape 2 by statement 200 of the main program. Whenever this terminal message is encountered, there is nothing left to be examined and REPORT terminates. Otherwise, if MSGE $\neq$ 0 then PRU is assigned the value

PRU = the number of the pru in the PFC to be reported on next

and GETPRU(PRU,I) is called. Next, the value of DMPNUM is increased by 1 and if DMPNUM $\leq$ DLIMIT a preface for a new dump and a copy of this preface are begun on tapes 1 and 10 respectively.

Currently, RBTCHK is not interested in doing a thorough analysis of the portion of a PFC entry that precedes the RB chain (if such an analysis is even possible). Instead, it checks only for those few errors, the existence of which leave in doubt either the validity of or the handling of the remaining data of the entry. When such an error is detected, the analysis of the pru is terminated and a tape 2 message is issued whose message number (MSGE) is less than 100. REPORT now treats these errors as follows. At statement 13 it is checked if the current tape 2 message refers to the pru under consideration If it does and if MSGE < 100, then either the pru is the first pru for a long PFC entry or

(1) the pru contains an EOR, EOF, or EOI mark,
(2) word 0 or 1 of the pru is bad,
(3) the number given in word 11 for the size of the PFC entry or the number given in word 12 which indicates where the RB chain for the entry begins is bad.

If (1),(2), or (3) occurs then at statement 110,120, or 122 CARRY is set to 0, a message is written on tapes 1 and 10 if DMPNUM $\leq$ DLIMIT, and the system proceeds to statement 200.[2]

If (1),(2),(3) do not occur, but the pru is the first pru for a long PFC entry, then at statement 100 CARRY is set to 1, and a comment is made on tapes 1 and 10 if DMPNUM $\leq$ DLIMIT. Thereupon, at statement 20 the name, cycle number, ID, and creation date of the file are obtained. Then the system arrives at statement 22. Otherwise, if the pru is not the first pru of a long PFC entry, then at the statement immediately following statement 13 REPORT checks if NWDS < I + 63; i.e., REPORT checks if the pru obtained in the second

---

[1] If DMPNUM $\leq$ DLIMIT then DMPNUM is also the number of dumps that have already been written on tape 1 by REPORT.

[2] The section of coding beginning at statement 100 and ending at the statement immediately preceding statement 200 is where all comments appearing in the prefaces of the dumps and in the copies of the prefaces are written.

reading of the PFC is short. It should not be short since case (1) is not involved here. However, if it is short then at statement 300 SIG becomes nonzero and the abort sequence of coding is entered.[1] Otherwise, if it is not short, then the verification of the data in the PFC is finished for the moment and it is next checked if the pru is the beginning pru of a PFC entry. If it is the beginning pru then CARRY is set to 0 and at statement 20 the name, cycle number, ID, and creation date of the file are obtained. However, if it is not the beginning pru then it must be a continuation pru for a long entry (prus not being used were ignored by RBTCHK). In this case, at statement 30 the value of CARRY is checked. Normally CARRY should be 1. If this is the case then we already have the name, cycle number, ID, and creation date for the file and the system proceeds to statement 22. Otherwise, if CARRY = 0 then the variable ID is set to 0 and again the system proceeds to statement 22. However, in this case the resetting of the variable ID indicates that the background information for the file is either not available or has already been given. It is not clear to the author whether this case can actually occur. However, to be on the safe side (and to protect ourselves in the eventuality that RBTCHK or REPORT is later modified) the situation is provided for by this coding.

At statement 22 REPORT begins its search for conflicts. First it checks if the pru under consideration is referenced in the CONPRU array. If the pru is not referenced then the system is instructed to go to statement 23. Otherwise, if the pru is referenced then there may be RB's in the pru that are in conflict for which no tape 2 messages were issued by RBTCHK. [At the time these RB's were first encountered, it was not known that they were also being used elsewhere.] For each such RB, an entry $ORD_1, RB_1, n, WD_1, \ldots, WD_n$ exists in the CON array where (ORD_1, RB_1) is the disk address of the RB and $SHIFT(PRU,6) = WD_1$. The GETCON routine is called for examining and reporting on the RB's in conflict for which there are no tape 2 messages. This routine is described in detail in the next section. After GETCON terminates, if SIG has been set to a nonzero value by GETCON then the system proceeds to statement 310, where the abort sequence of coding begins. Otherwise, if SIG = 0 then the value of T is increased by 1 and the system arrives at statement 23.

Statement 23 begins an inner loop. Each time that a pass is made through the loop a different tape 2 message for the pru under consideration is processed. The operation of the loop is quite simple. At statement 23 it is first checked if the next tape 2 message to be processed refers to the pru. If not, then the sytem exits from the loop and proceeds to statement 40. Otherwise, depending on whether the message refers to

(1) a bad RBR ordinal in the RB chain for the file,
(2) a nonexistent RB,
(3) an RB in conflict, or
(4) an error concerning the length of the RB chain

[1] If (1) occurs then at statement 110 it is similarly checked if $NWDS \geq I + 63$.

the system is instructed to go to statement 160,162,130,or 170. If (1),(2), or (4) occurs then CARRY is first set to 0. Thereupon, if DMPNUM ≤ DLIMIT then messages are issued to tapes 1 and 10 and the system proceeds to statement 200. However, if DMPNUM > DLIMIT then REPORT is finished with the pru. In this case, not only is the pass through the inner loop complete, but also the pass through the outer loop. The system now returns to statement 5 to begin the next pass through the outer loop.

If (3) occurs then at statement 130 W is assigned the value:

$$W = \begin{cases} 4 & \text{if the conflict involves a flawed RB} \\ 2 & \text{if the conflict is with the RB chain for the PFD} \\ 1 & \text{if the conflict is with the RB chain for the PFC} \\ 0 & \text{otherwise} \end{cases}$$

If W = 3, which should never occur, then at statement 302 SIG becomes nonzero and the abort sequence of coding begins. However, if W is 4, 2, or 1 then at statement 140 or 150 comments are first written on tapes 1 and 10 (whenever DMPNUM ≤ DLIMIT) and a new tape 2 message is read. Then the system returns to statement 23 to begin the next pass through the inner loop. Otherwise, if W = 0 then the NTRYCON routine is called. NTRYCON is the routine that provides background information to tape 3. Its definition

```
      SUBROUTINE NTRYCON(WD,RB,ORD)
      IMPLICIT INTEGER(A-Z)
      COMMON/TVAR/IBUFF1(260),PRU0,P,DLIMIT
      COMMON/RPRT/DMPNUM,FNAME(4),CYCLE,DATE,ID

      WRITE(3) WD,DMPNUM,FNAME,CYCLE,DATE,ID
      IF (DMPNUM.GT.DLIMIT) RETURN

      WORD=WD.AND.77B
      WRITE(1,10) RB,ORD,WORD
      WRITE(10,10) RB,ORD,WORD
10    FORMAT(*0WE HAVE A CONFLICT INVOLVING RB *,O4,* OF RBR ORDINAL *,
     XO2,*. THIS RB IS*/* REFERENCED IN WORD *,O2,*.*)
      RETURN
      END
```

requires no comment. After this subroutine ends, a new tape 2 message is read and the system returns to statement 23.

Statement 40 begins the section that prepares for the next pass through the (outer) loop. This section can only be entered at statements 40 and 41. It is entered at statement 40 only if there were no errors other than conflicts in the current pru. Otherwise, the system enters at statement 41 after a dump for the pru has been issued and the system is finished with the pru. At statement 40 it is first checked if DMPNUM ≤ DLIMIT. If this is the case then the system proceeds to statement 200, where a dump is to be written. [Later the system may re-enter this section at statement 41.] Otherwise, if DMPNUM > DLIMIT then

the value of the variable PRU is increased by 1 and GETPRU(PRU,I) obtains the next pru in the PFC. Then the system arrives at statement 41.

When statement 41 is reached, one of the following situations exists. First, the pru just examined and reported on may have contained a full PFC entry. If this is the case, then CARRY = 0 and the system is finished with the pru. Otherwise, if the pru contained a portion of a long PFC entry, then CARRY may have the value 0 or 1. If CARRY = 0 then the pru contained an error that left in doubt the validity of or the handling of the remainder of its data. Thus the system had no choice but to assume that the prus in the PFC immediately following this pru contained information not necessarily for this PFC entry, but for a different entry. However, if CARRY = 1 then the question remaining to be considered is whether the next pru to be reported on involves the current PFC entry or a different PFC entry. This question must now be answered.

At statement 41 it is assumed that the pru under consideration is the first pru following the pru just reported on, and that the next tape 2 message to be processed has already been read. First it is checked if CARRY = 0. If this is the case, then the current pass through the loop is complete and the system returns to statement 10. Otherwise, if CARRY = 1 and the next tape 2 message is the terminal message, then REPORT ends. However, if CARRY = 1 and the message is not the terminal message, then the system arrives at statement 42.

At statement 42 an inner loop is entered. Each time that a pass is made through this loop a different pru following the pru just reported on is checked. The prus are taken in sequence. At statement 42 it is first checked if the current pru under consideration is to be examined and reported on. If so, then the system exits from the inner loop, returning to statement 10 to begin the next pass through the outer loop. Otherwise, if the current pru is not to be reported on, then it is next checked if the pru is short. If it is short, then CARRY is set to 0 and the system proceeds to statement 10. Otherwise, at the statement following statement 43 it is checked if the pru is the beginning pru for a new PFC entry. If it is the beginning pru, then again CARRY is set to 0 and the system returns to statement 10. Otherwise, if it is not a beginning pru, then at statement 44 the value of the variable PRU is increased by 1 and GETPRU(PRU,I) obtains the next pru in the PFC. Thereupon, the system returns to statement 42 to begin the next pass through the inner loop.

Writing Dumps

Statement 200 begins the section for writing dumps. When this section is entered, it is assumed that all comments and/or complaints regarding the pru being reported on have been issued. The task now at hand is to provide a dump that contains this pru and its neighboring prus. It is required that the P prus immediately preceding and the Q prus immediately following this pru be

printed, if it is possible to do so. As noted earlier, P and Q may be arbitrarily set by the user, the only restriction being that $P \geq 2$ and $Q \geq 2$. The default values given at the beginning of REPORT are $P = 2$ and $Q = 2$.

At statement 200 it is first checked if the pru under consideration is the final pru of the PFC. If it is the final pru then the system proceeds to statement 202. Otherwise, if it is not the final pru, then the values of PRU and P are increased by 1, GETPRU(PRU,I) obtains the next pru in the PFC, and the system arrives at statement 202. Now let us stop and reflect on what has just been done. If the pru is not the final pru, then it must be noted that the value of the parameter P has been modified. If the initial value of P was p, then IBUFF1 may now hold the pru being reported on, as well as p previous prus. Thus IBUFF1 may contain a maximum of p+1 prus. Also, the variables PRU and I now refer to the pru immediately following the pru being reported on. After finishing with these prus, the variable P will have to be reset to its initial value p. The reader may breathe a sigh of relief, however. For it is only here that a (p+1)-st pru may be inserted into IBUFF1. In this case, because of the increase in the value of P, no information is lost in the insertion process. The queue IBUFF1 is momentarily treated as a stack.

Now to return from our detour, at statement 202 a line which separates the preface from the dump is written on tape 1, IRB is defined to be the number of the RB in the PFC that we currently are in, and K is assigned the value 0 or 1 depending on whether or not disk addresses are to be given for the initial prus in the dump.[1] Then P0 and J are assigned the values

```
P0 = 0
J = I - 64 * P
```

and it is checked if $J \geq 1$. If this is the case, then IBUFF1 is not needed and the system proceeds to statement 220. Otherwise, if $J < 1$, then it is next checked if the current pru is one of the first P - 1 prus of the PFC. If not, then the system proceeds to statement 210. Otherwise, if it is one of the beginning P - 1 prus, then only the prus that have been read can be printed. In this case, for convenience, only the prus in IBUFF are printed. P0 is assigned the value

```
P0 = 1 + (I - 1)/64
```

and the system proceeds to statement 221. This case can never occur if P is less than or equal to the number of prus containing the RB chain for the PFC. However, if the user redefines P to be greater than the number of prus containing the RB chain for the PFC, then the policy of printing only data in IBUFF may not be satisfactory. For then, if an EOR or EOF mark is being reported on, information that should be printed will not be printed.

Statement 210 begins the section for printing the prus in IBUFF1. First IPRU and P0 are assigned the values:

---

[1] PFCMRK and PFCRBS are defined by RBSET. For further information see pages 10 and 12.

IPRU = the number of the pru that precedes the first pru to be printed
PO = the number of prus in IBUFF1 to be printed

Then a DO loop is entered. Each time that a pass is made through the loop a different pru in IBUFF1 is written on tape 1. The prus are taken in sequence. At the beginning of a pass J has the value:

J = the index of the word in IBUFF1 that begins the entry of the pru to be printed next

First N is assigned to be the number of words in the pru and ICODE is set according to the format prescribed on page 17. Then the pru is written on tape 1 by the PRTPRU routine and the value of J is decreased by 65. This completes the pass through the loop. After the loop terminates, J is set to 1 and the system arrives at statement 220.

Statement 220 begins the section for writing the prus in IBUFF. After statement 221 has executed, IPRU and PO have the values:

IPRU = the number of the pru that precedes the first pru to be printed
PO = the number of prus in IBUFF (up to and including the current pru) that are to be printed

Then a DO loop is entered that writes the PO prus on tape 1. After this writing is finished then the value of P is decreased by 1, and the routine terminates if the pru being reported on is the final pru of the PFC. Otherwise, if the pru is not the final pru, then all the prus up to and including the first pru following the pru being reported on have been written and P has been reset to its initial value. Now the queue IBUFF1 has again its original size, and the remaining Q - 1 prus that follow the pru being reported on must be printed, if it is possible to do so. At statement 230 the next tape 2 message to be processed is obtained, and then the system proceeds to statement 240. (Currently, the READ statement following statement 230 can be encountered only once.)

The loop beginning at statement 240 is similar to the loop beginning at statement 42. Both loops prepare for the next pass through the outer loop. However, in the loop beginning at statement 240 a dump is still being written. At the beginning of a pass through this loop, it is first checked if the current pru is to be examined and reported on. (This pru has already been printed.) If so, then the system exits from the DO loop, preceding to statement 250. Otherwise, if the pru is not to be reported on, then it is next checked if the pru is short. If it is short, then CARRY is set to 0 and the system proceeds to statement 242. Otherwise, if it is not short, then at statement 241 it is first checked if CARRY = 0. If this is the case, then the system proceeds to statement 242. Otherwise, if CARRY $\neq$ 0 then it is next checked if the pru is the beginning pru for a new PFC entry. If not, then the system proceeds to statement 242. Otherwise, if it is the beginning pru, then CARRY is set to 0 and the system arrives at statement 242. At statement 242 the NXTPRU routine is called. NXTPRU(PRU,I,IND) obtains the next pru in the PFC and writes it on tape 1. Its definition

```
      SUBROUTINE NXTPRU(PRU,I,IND)
      IMPLICIT INTEGER(A-Z)
      COMMON/IFET/IBUFF(2049),RTCODE,NWDS,LEVEL,FILE
      COMMON/CHNVAR/PFDMRK,PFDRBS,PFDCHN(64),PFCMRK,PFCRBS,PFCCHN(320),
     X              MARK(3)

      PRU=PRU+1
      CALL GETPRU(PRU,I)
      IRB=1+PRU/56
      K=PFCMRK
      IF (IRB.GT.PFCRBS) K=1
      CALL PRTPRU(I,PRU,IBUFF(1),NWDS,RTCODE,K,IND)
      RETURN
      END
```

requires no comment. After NXTPRU finishes, it is checked if the new pru is the final pru of the PFC. If it is the final pru, then REPORT terminates. Otherwise, if it is not the final pru, then the system arrives at statement 243, which terminates the pass through the DO loop.

After leaving the DO loop, the system arrives at either statement 250 or the statement immediately following statement 243. If it arrives at the statement immediately following statement 243, then the Q prus following the pru being reported on have been printed and the system proceeds to statement 41, to continue the preparation for the next pass through the outer loop. Otherwise, if the system arrives at statement 250, then not all the Q prus following the pru being reported on can be included in the dump. The prus not included, however, can appear in the next dump. A comment to this effect is made on tape 1. This comment terminates the dump and the system proceeds to statement 10 to begin the next pass through the outer loop.

The Abort Section

Statement 310 begins the abort sequence of coding. Whenever SIG becomes nonzero, this section is entered. The action then taken is quite simple. First, messages are written on the output file and on tapes 1 and 10. Next, a list of the EOR, EOF, and EOI marks found by RBTCHK is provided by the loop beginning at statement 330. Then, at statement 340, the current values of SIG, I, and PRU0 are printed and the routine terminates. The list of EOR, EOF, and EOI marks may be of aid in determining if there is an error in the count of the prus. However, if there is no such error, then it is not clear what information should be given. The point to keep in mind is that REPORT does not continually monitor its progress. It only checks periodically for those few situations that it knows are absolutely fatal. Thus if an error does occur, then a considerable amount of time can elapse before the error is detected.

SUBROUTINE GETCON(IO,PRUNUM,SIG)

Let PRUNUM be the number of a pru in the PFC that is currently in IBUFF. Also let IO be the index of the word in IBUFF that begins this pru. Then the GETCON routine examines this pru for all the RB's in conflict for which there are no tape 2 messages. (At the time these RB's were first encountered, it was not known that they were being used elsewhere.) For each such RB, a conflict entry ORD,RB,n,WD_1,...,WD_n exists in the CON array where (ORD,RB) is the disk address of the RB and SHIFT(PRUNUM,6) = WD_1. Whenever GETCON finds such an RB, the RB is reported on and WD_1 is modified to contain the number of the word in the pru that first references this disk RB. The variable SIG serves the same purpose in this routine that it serves in REPORT.

The coding is quite simple. Statement 10 begins the outermost loop of the routine. Each time that a pass is made through the loop a different conflict entry in CON is examined. The entries are taken in sequence. At the beginning of a pass J has the value:

J = the index of the word in CON that follows the entry just examined

It is first checked if CON(J) begins a new entry. If not, then GETCON terminates. Otherwise, if it does begin a new entry, then it is next checked if SHIFT(PRUNUM,6) = CON(J + 3). If this is not the case, then at statement 11 J is updated and the sytem returns to statement 10, to begin the next pass through the outermost loop. Otherwise, at statement 20 I,WD,MAX are assigned the values:

MAX = the number of word pairs of the RB chain in pru PRUNUM
I = the index of the word in IBUFF where the RB chain begins
WD = the number of the word of the pru where the RB chain begins

Then the system proceeds to statement 100.

The DO loop beginning at statement 100 examines the word pairs of the RB chain that are in the pru. One of these word pairs must contain the RB in conflict that is referenced by the current conflict entry. However, if no word pair is found that contains the RB, then at statement 132 SIG is set to 1 and the routine terminates. The structure of the DO loop requires little comment, being the same as the structure of the corresponding loops in RBSET and RBTCHK. First the link of the word pair, the RBR ordinal ORD of the RB's referenced by the word pair, and the index IND of the first byte that references an RB are obtained. If the ordinal ORD is not the same as the ordinal CON(J) of the current conflict entry, then the system is finished with the word pair. Otherwise, if ORD = CON(J) then in the innermost loop beginning at statment 121 the RB bytes are checked. If no byte is found that contains the disk RB number CON(J + 1) of the current conflict entry, then again the system is finished with the word pair. Otherwise, if a byte is found that contains the disk RB number CON(J + 1), then CON(J + 3) is modified to contain the number of the word in the pru that contains this byte. Also NTRYCON[1] is called to report on the conflict, and then the system

[1] See page 33.

returns to statement 11 to finish this pass through the outermost loop.

Note The variables I,WD,MAX should not have been defined in the outermost loop. Instead, for efficiency, the variables should have been defined at the beginning of the routine.

SUBROUTINE CONTBL

After REPORT has ended, the PFC is closed by PFRETRN, an entry consisting entirely of zeros is written on tape 3, and tape 3 is rewound. Then the CONTBL routine is called. This routine and its subroutine PRTDATA print a table which cross references the conflicts not involving flawed RB's or RB's in the RB chains for the PFD and PFC. After CONTBL ends, tapes 10 and 1 are printed and the main program terminates.

The CON array contains the cross tabulation in abbreviated form. Statement 10 begins the major loop of the routine. Each time that a pass is made through the loop a different conflict entry ORD,RB,n,WD_1,...,WD_n is reported on. The entries are taken in sequence. After all the entries have been processed, then the routine ends.

The PRTDATA routine is called by CONTBL to print background information on each of the words WD_i in the conflict entries. Included in this information is the name of the file that contains the word, the cycle number, ID, and creation date of the file, and the number of the dump which contains the word. When PRTDATA(WD_i) is called, tape 3 is searched for an entry written by the NTRYCON routine that references word WD_i. If an entry is found then the information provided by the entry is printed. If the ID of the entry is 0, then this indicates that only the number of the dump that contains this word is available. [1]

The coding of PRTDATA is fairly straightforward, requiring only the use of the IBLANK function [2] and the GETDATE routine. GETDATE has the definition:

```
      SUBROUTINE GETDATE(LP,N,M,D)
      INTEGER D
C     THIS ROUTINE COMPUTES THE MONTH (M) AND THE DAY (D) OF THE MONTH
C     FOR THE N-TH DAY OF THE YEAR. (IF THE YEAR IS A LEAP YEAR THEN N
C     CAN HAVE ANY OF THE VALUES 1,2,...,366.)
C     LP=1 IF THE YEAR IS A LEAP YEAR. OTHERWISE LP=0.
      M=1
      D=N
      IF (D.LE.(59+LP)) GO TO 11
      M=3
      D=D-59-LP
      IF (D.LE.153) GO TO 10
      M=8
      D=D-153
10    K=(D-1)/61
      M=M+2*K
      D=D-61*K
```

[1] See page 32.
[2] See page 18.

```
11    IF (D.LE.31) RETURN
      M=M+1
      D=D-31
      RETURN
      END
```

This coding may appear a bit enigmatical. However, if one realizes that the last 10 months of the year divide evenly into 2 groups, each of which has 5 months containing 31,30,31,30,31 days respectively, then the coding should begin to come into focus.

References

1. SCOPE System Programmer's Reference Manual (Models 72, 73, 74 Version 3.4, 6000 Version 3.4). Publication No. 60306500, Control Data Corporation, Sunnyvale, Calif., 1974.

2. SCOPE Reference Manual (Models 72, 73, 74 Version 3.4, 6000 Version 3.4). Publication No. 60307200, Control Data Corporation, Sunnyvale, Calif., 1974.

APPENDIX A

LISTINGS

```
      OVERLAY(RBTC,0,0)
      PROGRAM RBTC(SYSTEM=0,OUTPUT,TAPE1,TAPE2,TAPE3,TAPE10)
      IMPLICIT INTEGER(A-Z)
      COMMON/IFET/IBUFF(2049),RTCODE,NWDS,LEVEL,FILE
      COMMON RBRTBL(3232,8),AREA(3),ENTRY(3),NUMTBL
      COMMON/RBVAR/MCOUNT,WD,I,IERR
      COMMON/CHNVAR/PFDMRK,PFDRBS,PFDCHN(64),PFCMRK,PFCRBS,PFCCHN(320),
     X              MARK(3)
      INTEGER LINE(14)

CCC                              *********
C
C                 RB CHAIN/RBTC ANALYZER --- VERSION 1.0
C                           DATE -- DEC 1975
C
C                        ALFRED H. MORRIS, JR.
C                        CODE DK74
C                        NAVAL SURFACE WEAPONS CENTER
C                        DAHLGREN, VIRGINIA 22448
C
C     THIS PROGRAM CHECKS THE RB CHAINS OF THE PFD AND PFC, AND EXAMINES
C     THE RB C ENTRIES (INCLUDING THEIR RB CHAINS). IT IS ASSUMED THAT
C          (1) THE RBTC IS NOT LONGER THAN 1170 (DECIMAL) RBS
C              IN LENGTH,
C          (2) THE LARGEST RBR ORDINAL THAT MAY BE USED BY THE
C              PERMANENT FILES IS 39,
C          (3) AT ANY ONE TIME NO MORE THAN 24 OF THE POSSIBLE
C              40 RBR ORDINALS ARE BEING USED,
C          (4) THERE MAY ONLY BE A MAXIMUM OF 3232 (DECIMAL) RBS
C              FOR EACH RBR ORDINAL, AND
C          (5) THERE IS ONLY ONE RBR ORDINAL FOR EACH PERMANENT
C              FILE DEVICE.
C     THIS ANALYZER IS FOR THE SCOPE 3.4.3 OPERATING SYSTEM.
C
C                              *********
C
C
C     THE CURRENT VALUES FOR THE VARIABLES NUMTBL,MCOUNT,PFDSIZE,PFCSIZE
C     ARE ...
C
C     NUMTBL=39
C     MCOUNT=4000
C     PFDSIZE=1
C     PFCSIZE=5
C
C     THESE VALUES MAY BE CHANGED BY THE USER IF HE WISHES.
C
C     NUMTBL IS THE LARGEST RBR ORDINAL USED. IT CANNOT EXCEED 39.
C
C     MCOUNT IS THE MAXIMUM NUMBER OF PECULARITIES IN THE RBTC ENTRIES
C     THAT WILL BE REPORTED ON. MCOUNT CANNOT EXCEED 4000. (FOR FURTHER
C     DETAILS CONCERNING MCOUNT SEE THE COMMENTS AT THE BEGINNING OF THE
C     RBTCHK AND INFOCON ROUTINES.)
C
C     PFDSIZE = THE MAXIMUM NUMBER OF PRUS THAT THE RB CHAIN FOR THE
C               PFD SHOULD TAKE
C
C     PFCSIZE = THE MAXIMUM NUMBER OF PRUS THAT THE RB CHAIN FOR THE
```

```
C                PFC SHOULD TAKE
C
C      THE USER MAY ALSO CHANGE (AT THE BEGINNING OF THE REPORT ROUTINE)
C      THE VALUES OF THE VARIABLES P,Q,DLIMIT. FOR FURTHER DETAILS SEE
C      THIS ROUTINE.
C
C                                  *********

C      THE GETFILE ROUTINE CREATES THE FNT ENTRY FOR THE FILE REQUESTED,
C      AND THE TEST ROUTINE CREATES THE FET THE FIRST TIME THAT IT IS
C      CALLED. THE PFRETRN ROUTINE ELIMINATES THE FNT AND FET ENTRIES FOR
C      THIS FILE. THE FILE IS REFERENCED BY SETTING THE VARIABLE FILE.
C           FILE=0 IF THE PFD IS THE FILE TO BE READ
C           FILE=1 IF THE RBTC IS THE FILE TO BE READ
C      EACH TIME TEST IS CALLED, TEST READS A PORTION OF THE FILE, STORES
C      IT IN THE IBUFF BLOCK, AND SETS THE VARIABLES RTCODE,NWDS,LEVEL.
C           RTCODE=1 IF AN EOR MARK IS READ
C           RTCODE=2 IF AN EOF MARK IS READ
C           RTCODE=3 IF AN EOI MARK IS READ
C      OTHERWISE RTCODE IS SET TO 0. ALSO
C           NWDS = THE NUMBER OF WORDS READ INTO IBUFF
C           LEVEL = THE LEVEL NUMBER OF THE RECORD
C      A MAXIMUM OF 2048 WORDS IS READ. IF AN EOR, EOF, OR EOI MARK IS
C      ENCOUNTERED, THEN READING TERMINATES. IN THIS CASE NWDS IS THE
C      NUMBER OF WORDS READ THAT PRECEDE THE MARK.

C                                  *********

       NUMTBL=39
       MCOUNT=4000
       PFDSIZE=1
       PFCSIZE=5

       IZERO=0
       NCOUNT=MCOUNT
       MARK(1)=0517223
       MARK(2)=0517063
       MARK(3)=0517115

       PRINT 10
10     FORMAT(*1COMMENTS ...*/)
       PRINT 11
11     FORMAT(*0UNLESS IT IS CLEAR TO THE CONTRARY, ANY NUMBER APPEARING
      XIN THE OUTPUT*)
       PRINT 12
12     FORMAT(* CAN BE ASSUMED TO BE IN OCTAL.*)

       CALL SETUP(ERROR)
       IF (ERROR.NE.0) STOP
       FILE=0
       CALL GETFILE
       WD=4000008
       CALL RBSET(PFDSIZE,PFDMRK,PFDRBS,PFDCHN(1))
       CALL PFRETRN
       IF (WD.EQ.0) GO TO 300

       FILE=1
```

```
      CALL GETFILE
      WD=2000008
      CALL RBSET(PFCSIZE,PFCMRK,PFCRBS,PFCCHN(1))
      IF (WD.NE.0) GO TO 100
      CALL PFRETRN
      GO TO 300

100   IERR=PFCMRK*PFCMRK
      WD=WD.AND.1777778
      CALL RBTCHK
      CALL PFRETRN
      MCOUNT=NCOUNT-MCOUNT

      CALL REMARK(10H**********)
      IF (IERR.EQ.0) GO TO 111
      CALL DISPLA(42HNUMBER OF ERRORS THAT ARE NOT CONFLICTS = ,IERR)
      PRINT 110,IERR
110   FORMAT(*0NUMBER OF ERRORS DETECTED THAT ARE NOT CONFLICTS =*,I4,
     X* (DECIMAL).*)
      GO TO 200
111   CALL REMARK(48HNO ERRORS (OTHER THAN POSSIBLY CONFLICTS) FOUND.)
      PRINT 112
112   FORMAT(*0NO ERRORS (OTHER THAN POSSIBLY CONFLICTS) FOUND.*)

200   WRITE(2,201) IZERO,IZERO,IZERO,IZERO,IZERO
201   FORMAT(5020)
      REWIND 2
      CALL INFOCON(MCOUNT)
      CALL REMARK(10H**********)

      FILE=1
      CALL GETFILE
      CALL REPORT
      CALL PFRETRN

      WRITE(3) IZERO,IZERO,IZERO,IZERO,IZERO,IZERO,IZERO,IZERO,IZERO
      REWIND 3
      CALL CONTBL

300   REWIND 10
301   READ(10,302) LINE
302   FORMAT(13A10,A6)
      IF (EOF(10).NE.0) GO TO 310
      PRINT 302,LINE
      GO TO 301

310   REWIND 1
311   READ(1,302) LINE
      IF (EOF(1).NE.0) STOP
      PRINT 302,LINE
      GO TO 311
      END
```

```
      SUBROUTINE SETUP(ERR)
      IMPLICIT INTEGER(A-Z)
      COMMON RBRTBL(3232,8),AREA(3),ENTRY(3),NUMTBL
      INTEGER TABLE(25856),FLAW(3)
      EQUIVALENCE (RBRTBL(1,1),TABLE(1))
      COMMON/RBRORD/MORD(40),NORD(40)
      INTEGER LEST(64),LABEL(128)

C     LEST = THE LIST OF EST ORDINALS FOR THE PERMANENT FILE DEVICES
C            THAT ARE TURNED ON
C     NUM = THE NUMBER OF PERMANENT FILE DEVICES THAT ARE TURNED ON

C     THE LABEL ARRAY CONTAINS THE LABEL FOR THE PERMANENT FILE DEVICE.
C     THE RB BITS IN THE LABEL HAVE BEEN REARRANGED SO THAT THEY APPEAR
C     CONSECUTIVELY IN WORDS 11-64 OF THE ARRAY. IT IS ASSUMED THAT
C     THERE IS ONLY ONE RBR ORDINAL FOR EACH PERMANENT FILE DEVICE.

      AREA(1)=777777600000030000008
      AREA(2)=000001777777400000B
      AREA(3)=000000000000037777778
      FLAW(1)=200000000000000000008
      FLAW(2)=000003040000030000008
      FLAW(3)=000000000000010000008

      DO 10 II=1,25856
10    TABLE(II)=0
      DO 11 II=1,40
11    MORD(II)=NORD(II)=0

      M=0
      N=1
      CALL RBEST(NUM,LEST)
      IF (NUM.LE.0) GO TO 100
      IF (NUM.GT.24) GO TO 103

      DO 32 II=1,NUM
      EST=LEST(II)
      CALL RLABEL(EST,LABEL,ERR)
      IF (ERR.NE.0) GO TO 110
      K=SHIFT(LABEL(10),-24).AND.777B
      IF (K.NE.EST) GO TO 112

      K=LABEL(10).AND.77B
      IF (K.GT.NUMTBL) GO TO 120
      IF (M.LT.3) GO TO 20
      M=0
      N=N+1
20    M=M+1
      MORD(10*M+1)=M
      NORD(10*M+1)=N

C     HERE M=1,2,3 AND N=1,2,...,8.

      MAX=LABEL(10).AND.77777B
      IF (MAX.EQ.0) GO TO 130
      IF (MAX.GT.3232) GO TO 122
```

```
      I=11
      RB=1
30    W=LABEL(I)
      DO 31 JJ=1,60
      W=SHIFT(W,1)
      IF ((W.AND.1).NE.0) RBRTBL(RB,N)=FLAW(M).OR.RBRTBL(RB,N)
      IF (RB.EQ.MAX) GO TO 32
31    RB=RB+1
      I=I+1
      GO TO 30
32    CONTINUE
      RETURN

C                     ERROR PROCESSOR

100   PRINT 101
101   FORMAT(5H0****,*THE JOB WAS ABORTED BEFORE PROCESSING EVEN BEGAN.
     XTHE REASON FOR THIS ABORT WAS ...*)
      PRINT 102
102   FORMAT(*0NO PERMANENT FILE DEVICES ARE LISTED AS BEING ON.*)
      ERR=1
      RETURN
103   PRINT 101
      PRINT 104
104   FORMAT(*0MORE THAN 24 PERMANENT FILE DEVICES ARE LISTED AS BEING O
     XN. ONLY 24 ARE PERMITTED*/* BY THIS VERSION OF THE ANALYZER.*)
      ERR=1
      RETURN

110   PRINT 101
      PRINT 111,ERR,EST
111   FORMAT(*0ERROR *,I2,* (DECIMAL) OCCURRED WHEN THE PP ROUTINE TAT W
     XAS READING THE DEVICE*/* HAVING EST ORDINAL *,02,*.*)
      RETURN
112   PRINT 101
      PRINT 113,K,EST
113   FORMAT(*0ONE OF THE PERMANENT FILE PACKS IS NOT MOUNTED ON THE COR
     1RECT EST ORDINAL. THE CORRECT ORDINAL*/* IS *,02,*. THE INCORRECT
     2ORDINAL IS *,02,*.*)
      ERR=1
      RETURN

120   PRINT 101
      PRINT 121,ORD
121   FORMAT(*0PBR ORDINAL *,I2,* (DECIMAL) IS BEING USED. THE LARGEST R
     1BR ORDINAL THAT IS PERMITTED*/* BY THIS VERSION OF THE ANALYZER IS
     2 39.*)
      ERR=1
      RETURN

130   PRINT 101
      PRINT 131,EST
131   FORMAT(*0NO USABLE RBS ARE ON THE PERMANENT FILE DEVICE HAVING EST
     X ORDINAL *,02,*.*)
      ERR=1
      RETURN
132   PRINT 101
```

```
      PRINT 133,EST
133   FORMAT(*0MORE THAN 3232 (DECIMAL) PBS ARE GIVEN FOR THE PERMANENT
     1FILE DEVICE HAVING EST ORDINAL *,02,*.*/* THIS IS NOT PERMITTED BY
     2 THIS VERSION OF THE ANALYZER.*)
      ERR=1
      RETURN
      END
```

```
      SUBROUTINE RBSET(CHSIZE,NUM,DSKRBS,CHAIN)
      IMPLICIT INTEGER(A-Z)
      COMMON/IFET/IBUFF(2049),RTCODE,NWDS,LEVEL,FILE
      COMMON RBRTBL(3232,8),AREA(3),ENTRY(3),NUMTBL
      COMMON/RBVAR/MCOUNT,WD,I,IERR
      COMMON/RBRORD/MORD(40),NORD(40)
      COMMON/CHNVAR/PFDMRK,PFDRBS,PFDCHN(64),PFCMRK,PFCRBS,PFCCHN(320),
     X              MARK(3)
      INTEGER CHAIN(1)

C     IN THIS ROUTINE IBUFF CONTAINS THE BEGINNING PRUS OF THE PFD OR
C     PFC. THE RB CHAIN FOR THE PFD OR PFC IS STORED IN THESE PRUS.
C     THIS ROUTINE EXAMINES THE RB CHAIN. IF ANY EOR OR EOF MARKS ARE
C     FOUND TO PRECEDE THE RB CHAIN, THE ERROR IS REPORTED AND ANALYSIS
C     IS CONTINUED. IF ANY OTHER ERRORS ARE DETECTED, THEN ANALYSIS IS
C     TERMINATED AND A DUMP OF THE RB CHAIN IS GIVEN. IN THIS CASE WD
C     IS SET TO 0, WHICH INDICATES TO THE MAIN PROGRAM TO TERMINATE
C     THE JOB. IF NO OTHER ERRORS ARE DETECTED, THE ROUTINE STORES THE
C     RB CHAIN FOR THE PFD IN THE ARRAY PFDCHN AND THE RB CHAIN FOR THE
C     PFC IN THE ARRAY PFCCHN.

C     PFDMRK = THE NUMBER OF EOR AND/OR EOF MARKS PRECEDING THE RB
C              CHAIN FOR THE PFD
C     PFCMRK = THE NUMBER OF EOR AND/OR EOF MARKS PRECEDING THE RB
C              CHAIN FOR THE PFC

C     PFDRBS = THE NUMBER OF DISK RBS IN THE RB CHAIN FOR THE PFD
C     PFCRBS = THE NUMBER OF DISK RBS IN THE RB CHAIN FOR THE PFC

C     UNLESS AN ERROR HAS OCCURRED, PFDMRK AND PFCMRK SHOULD BOTH HAVE
C     THE VALUE 0.

C     CHSIZE = THE NUMBER OF PRUS IN THE PFD OR PFC THAT ARE REQUIRED
C              FOR STORING THE RB CHAIN FOR THE PFD OR PFC

C     WD = MU + THE NUMBER OF WORDS IN THE PFD OR PFC THAT HAVE BEEN
C          PROCESSED
C     I = THE INDEX OF THE WORD IN IBUFF TO BE EXAMINED NEXT

C     HERE IT IS ASSUMED THAT NEITHER THE PFD NOR THE PFC HAS AN RB
C     CHAIN THAT CONTAINS MORE THAN 2**16 WORDS. MU=4000000B IF THE PFD
C     IS BEING READ AND MU=2000000B IF THE PFC IS BEING READ.

C     RBRTBL AND NUMTBL ARE DEFINED IN THE ROUTINE RBTCHK.

      I=1
      NUM=0
      DSKRBS=0
      CALL TEST

10    IF (NWDS.GT.0) GO TO 20
      IF (RTCODE.EQ.3) GO TO 250
      WD=WD+64
      NUM=NUM+1
      IF (NUM.EQ.50) GO TO 252
      CALL TEST
      GO TO 10
```

```
20      IF (NUM.EQ.0) GO TO 100
        K=SHIFT(WD,-17)+3
        PRINT 21,K,NUM,K
21      FORMAT(*0THE RB CHAIN FOR THE PF*,R1,* IS PRECEDED BY *,O2,* EOR A
       1ND/OR EOF MARKS. THE*/* ERROR MUST BE CORRECTED. ANALYSIS WAS CONT
       2INUED AFTER THIS ERROR WAS*/* DETECTED. HOWEVER, BECAUSE OF THE ER
       3ROR NO DISK ADDRESSES ARE GIVEN*/* FOR THE WORDS IN THE PF*,R1,
       4*.*)


C       EXAMINING THE RB CHAIN FOR THE RFD OR THE RBTC

100     MAXWPR=32*CHSIZE
        DO 131 II=1,MAXWPR
        IF (I.GE.NWDS) GO TO 240
        LINK=IBUFF(I).AND.777700000000000000008
        W=SHIFT(IBUFF(I),24)
        IND=W.AND.7
        ORD=W.AND.7770B
        ORD=SHIFT(ORD,-3)
        IF (ORD.GT.NUMTBL) GO TO 200
        M=MORD(ORD+1)
        N=NORD(ORD+1)
        IF (N.EQ.0) GO TO 200

C       HERE M=1,2,3 AND N=1,2,...,8.

        IF (IND.EQ.0) GO TO 110
        IF (IND.EQ.3) GO TO 120
        CALL ADDRSS(WD,K1,NUMRB,PRU,W1)
        PRINT 101,IBUFF(I),IBUFF(I+1)
101     FORMAT(*0THE RB BYTE INDICATOR (BITS 36-38 OF THE FIRST WORD) OF T
       XHE RB CHAIN*/* WORD PAIR*,2022,*  IS NOT 0 OR 3,*)
        PRINT 102,IND,NUMRB,PRU,W1,K1
102     FORMAT(* BUT *,O1,*. THIS WORD PAIR BEGINS AT RB *,O4,* PRU *,O2,
       X* WORD *,O2,* OF THE RB CHAIN*/* FOR THE PF*,R1,*.*)
        IF (IND-3) 111,111,112

110     K=1
        IND=5
        GO TO 121
111     K=1
        W=SHIFT(W,IND*12)
        IND=IND+5
        GO TO 121
112     K=2
        I=I+1
        WD=WD+1
        W=SHIFT(IBUFF(I),(IND-3)*12)
        GO TO 121

120     K=2
        I=I+1
        WD=WD+1
        W=IBUFF(I)
121     ENTRY(3)=WD
```

```
      ENTRY(2)=SHIFT(ENTRY(3),20)
      ENTRY(1)=SHIFT(ENTRY(2),20)
      DO 122 J=IND,7
      W=SHIFT(W,12)
      RB=W.AND.7777B
      IF (RB.EQ.0) GO TO (123,130),K
      IF (RB.GT.3232) GO TO 202
      INFO=AREA(M).AND.RBRTBL(RB,N)
      IF (INFO.NE.0) GO TO 210
      DSKRBS=DSKRBS+1
122   RBRTBL(RB,N)=ENTRY(M).OR.RBRTBL(RB,N)
      IND=3
      GO TO (120,130),K
123   I=I+1
      WD=WD+1

130   IF (LINK.EQ.0) GO TO 140
      I=I+1
131   WD=WD+1
      GO TO 242

140   DO 141 KK=1,I
141   CHAIN(KK)=IBUFF(KK)
      W=WD.AND.77B
      WD=WD+64-W
      I=I+64-W
      IF ((I.GT.NWDS).AND.(RTCODE.EQ.3)) GO TO 260
      IF ((NUM.EQ.0).AND.(I.LE.NWDS)) RETURN

      K=SHIFT(WD,-17)+3
      WRITE(1,150) K
150   FORMAT(*1RB CHAIN FOR THE PF*,P1,*...*/)
      J=1
      W=NUM
151   CALL RDTPRU(J,W,IBUFF(1),NWDS,RTCODE,1,IND)
      J=J+64
      W=W+1
      IF (J.LT.I) GO TO 151

      IF (I.LE.NWDS) RETURN
      CALL TEST
      I=1
      RETURN

C                    ERROR PROCESSOR

200   CALL ADDRSS(WD,K,NUMRB,PRU,W)
      PRINT 201,NUMRB,PRU,W,K
201   FORMAT(*1THE RBR ORGINAL IN RB *,O4,* PRU *,O2,* WORD *,O2,* OF TH
     XE RB CHAIN FOR THE PF*,P1/* IS BAD.*)
      GO TO 206
202   CALL ADDRSS(WD,K,NUMRB,PRU,W)
      PRINT 203,RB,NUMRB,PRU,W,K
203   FORMAT(*1THE VALUE *,O4,* IN RB *,O4,* PRU *,O2,* WORD *,O2,* OF T
     XHE RB CHAIN FOR THE PF*,P1/* IS BAD.*)
206   PRINT 207
207   FORMAT(*0THE ERROR MUST BE CORRECTED. ANALYSIS WAS TERMINATED WHEN
```

```
     X THIS ERROR*/* WAS DETECTED.*)
208   IF ((NUM.NE.0).OR.(NUMRB.GT.DSKRBS)) GO TO 300
      CALL DSKADD(NUMRB,IBUFF(1),ORD,RB)
      PRINT 209,NUMRB,K,RB,ORD
209   FORMAT(*0THE DISK ADDRESS OF RB *,O4,* OF THE RB CHAIN FOR THE PF*
     X,R1,* IS RB *,O4,* OF*/* RBR ORDINAL *,O2,*.*)
      GO TO 300


210   CALL ADDRSS(WD,K,NUMRB,PRU,W)
      INFO=SHIFT(INFO,20*M)
      PRINT 211,RB,ORD
211   FORMAT(*1WE HAVE A FATAL CONFLICT INVOLVING RB *,O4,* OF RBR ORDIN
     XAL *,O2,*. THIS RB*)
      IF (SHIFT(INFO,-18).EQ.0) GO TO 220
      PRINT 212,NUMRB,PRU
212   FORMAT(* IS FLAWED (AND HENCE NOT USABLE), BUT IS REFERENCED IN RB
     X *,O4,* PRU *,O2)
      PRINT 213,W,K
213   FORMAT(* WORD *,O2,* OF THE RB CHAIN FOR THE PF*,R1,*.*)
      GO TO 206


220   CALL ADDRSS(INFO,K1,NUMRB1,PRU1,W1)
      IF (K.EQ.K1) GO TO 230
      PRINT 221,NUMRB1,PRU1,W1
221   FORMAT(* IS REFERENCED IN RB *,O4,* PRU *,O2,* WORD *,O2,* OF THE
     XRB CHAIN FOR THE PFD*)
      PRINT 222,NUMRB,PRU,W
222   FORMAT(* AND IN RB *,O4,* PRU *,O2,* WORD *,O2,* OF THE RB CHAIN F
     XOR THE RBTC.*)
      PRINT 207
      IF ((PFDMRK.NE.0).OR.(NUMRB1.GT.PFDRBS)) GO TO 208
      CALL DSKADD(NUMRB1,PFDCHN(1),ORD,RB)
      PRINT 209,NUMRB1,K1,RB,ORD
      GO TO 208


230   PRINT 231,NUMRB1,PRU1,W1,NUMRB,PRU,W,K
231   FORMAT(* IS REFERENCED IN RB *,O4,* PRU *,O2,* WORD *,O2,* AND RB
     1*,O4,* PRU *,O2,* WORD *,O2,* OF*/* THE RB CHAIN FOR THE PF*,R1,
     2*.*)
      PRINT 207
      IF ((NUM.NE.0).OR.(NUMRB.GT.DSKRBS)) GO TO 300
      CALL DSKADD(NUMRB1,IBUFF(1),ORD,RB)
      PRINT 232,NUMRB1,RB,ORD
232   FORMAT(*0THE DISK ADDRESS OF RB *,O4,* IS RB *,O4,* OF RBR ORDINAL
     X *,O2,*, AND THE*)
      CALL DSKADD(NUMRB,IBUFF(1),ORD,RB)
      PRINT 233,NUMRB,RB,ORD
233   FORMAT(* DISK ADDRESS OF RB *,O4,* IS RB *,O4,* OF RBR ORDINAL *,
     XO2,*.*)
      GO TO 300


240   IF (I.E).NWDS) WD=WD+1
      CALL ADDRSS(WD,K,NUMRB,PRU,W)
```

```
      PRINT 241,MARK(RTCODE),NUMRB,PRU,W,K
241   FORMAT(*1AN *,R3,* MARK OCCURS AT RB *,04,* PRU *,02,* WORD *,02,
     X* IN THE MIDDLE OF THE RB*/* CHAIN FOR THE PF*,R1,*.*)
      GO TO 206
242   K=SHIFT(WD,-17)+3
      PRINT 243,K,CHSIZE
243   FORMAT(*1THE RB CHAIN FOR THE PF*,R1,* TAKES MORE THAN *,02,
     X* PRUS OF STORAGE.*)
      PRINT 207
      GO TO 300


250   CALL ADDRSS(WD,K,NUMRB,PRU,W)
      PRINT 251,K,NUMRB,PRU,W
251   FORMAT(*1THE RB CHAIN FOR THE PF*,R1,* IS PRECEDED BY AN EOI MARK
     XAT RB *,04,* PRU *,02/* WORD *,02,*.*)
      PRINT 207
      WD=0
      RETURN
252   K=SHIFT(WD,-17)+3
      PRINT 253,K,NUM
253   FORMAT(*1THE RB CHAIN FOR THE PF*,R1,* IS PRECEDED BY *,02,* ECR A
     XND/OR EOF MARKS.*)
      PRINT 207
      WD=0
      RETURN


260   I=I-NWDS-1
      WD=WD-I
      CALL ADDRSS(WD,K,NUMRB,PRU,W)
      PRINT 261,K,NUMRB,PRU,W
261   FORMAT(*1THE RB CHAIN FOR THE PF*,R1,* IS FOLLOWED BY AN EOI MARK
     XAT RB *,04,* PRU *,02/* WORD *,02,*.*)
      GO TO 206


C     PRINTING THE RB CHAIN AND SEVERAL OF THE FOLLOWING NONZERO PRUS

300   I=1
      W=NUM
      N=CHSIZE+3
      K=SHIFT(WD,-17)+3
      WRITE(1,150) K
      WD=0

310   CALL PRTPRU(I,W,IBUFF(1),NWDS,RTCODE,1,IND)
      I=I+64
      W=W+1
      N=N-1
      IF (N.EQ.0) RETURN

      IF (IND.EQ.0) GO TO 310
      IF (RTCODE.EQ.3) RETURN
      CALL TEST
      I=1
      GO TO 310
```

```
      END

      SUBROUTINE ADDRSS(M,I,RB,PRU,WORD)
      INTEGER RB,PRU,WORD

C     M IS A 20 BIT POSITIVE INTEGER (RIGHT ADJUSTED) THAT REFERENCES
C     A WORD IN THE RB CHAIN FOR THE RBTC OR PFD. THIS ROUTINE DETER-
C     MINES THE ADDRESS (RB,PRU,WORD) OF THE WORD. I IS SET TO 3 IF
C     THE WORD IS IN THE RB CHAIN FOR THE PFC, AND I IS SET TO 4 IF
C     THE WORD IS IN THE RB CHAIN FOR THE PFD.

      I=SHIFT(M,-17)+3
      WORD=M.AND.77B
      PRU=SHIFT(M,-6).AND.1777B
      RB=PRU/56
      PRU=PRU-RB*56
      RB=RB+1
      END

      SUBROUTINE DSKADD(NUMRB,CHAIN,ORD,RB)
      INTEGER CHAIN(1),ORD,RB,W

C     CHAIN = THE RB CHAIN FOR THE RBTC OR PFD
C     NUMRB = THE NUMBER OF AN RB IN THE RBTC OR PFD

C     THE ROUTINE FINDS THE RBP ORDINAL (ORD) AND THE DISK RB NUMBER
C     (RB) FOR THE RB REFERENCED BY THE NUMBER NUMRB.

      I=1
      NUM=0

100   W=SHIFT(CHAIN(I),24)
      IND=W.AND.7
      ORD=W.AND.7770B
      ORD=SHIFT(ORD,-3)
      IF (IND.EQ.0) GO TO 110
      IF (IND-3) 111,120,112

110   K=1
      IND=5
      GO TO 121
111   K=1
      W=SHIFT(W,IND*12)
      IND=IND+5
      GO TO 121
112   K=2
      I=I+1
      W=SHIFT(CHAIN(I),(IND-3)*12)
      GO TO 121

120   K=2
      I=I+1
      W=CHAIN(I)
121   DO 122 J=IND,7
      W=SHIFT(W,12)
      RB=W.AND.7777B
      IF (RB.EQ.0) GO TO (123,130),K
```

```
      NUM=NUM+1
      IF (NUM.EQ.NUMRB) RETURN
122   CONTINUE
      IND=3
      GO TO (120,130),K
123   I=I+1

130   I=I+1
      GO TO 100
      END
```

```
      SUBROUTINE PRTPRU(I,PRU,MEM,NUMWDS,ICODE,K,IND)
      IMPLICIT INTEGER(A-Z)
      COMMON/CHNVAR/PFDMRK,PFDRBS,PFDCHN(64),PFCMRK,PFCRBS,PFCCHN(320),
     X              MARK(3)
      DIMENSION MEM(1)

C     A PRU IN THE ARRAY MEM IS TO BE PRINTED. THE DISK ADDRESS IS
C     PRINTED IF K=0. OTHERWISE, IF K IS NOT 0 THEN THE DISK ADDRESS
C     IS NOT PRINTED. IF THE PRU CONTAINS 64 WORDS THEN THE INDICATOR
C     IND IS SET TO 0. OTHERWISE IND IS SET TO 1.

C     I = THE INDEX OF THE FIRST WORD OF THE PRU IN MEM
C     PRU = THE NUMBER OF THE PRU
C     NUMWDS = THE NUMBER OF WORDS IN MEM

C     IF I IS GREATER THAN NUMWDS THEN IT IS ASSUMED THAT I=NUMWDS+1.
C     IN THIS CASE THE PRU IS ASSUMED TO CONTAIN NO WORDS. ALSO, IF
C     MEM=IBUFF THEN I IS ASSUMED NOT TO BE 2049.

      WORD=0
      MAX=I+63
      IF (MAX.LE.NUMWDS) GO TO 10
      IND=1
      MAX=NUMWDS
      GO TO 20

10    IND=0
      DO 11 J=1,64
      IF (MEM(MAX).NE.0) GO TO 20
11    MAX=MAX-1

20    NUMRB=PRU/56
      NUMPRU=PRU-NUMRB*56
      NUMRB=NUMRB+1
      IF (K) 21,23
21    WRITE(1,22) NUMRB,NUMPRU
22    FORMAT(*0*,T50,*RB *,O4,* PRU *,O2)
      GO TO 30
23    CALL DSKADD(NUMRB,PFCCHN(1),IORD,IRB)
      WRITE(1,24) NUMRB,NUMPRU,IRB,IORD
24    FORMAT(*0*,T38,*RB *,O4,* PRU *,O2,* (DISK RB *,O4,*/RBR ORDINAL *
     X,O2,*)*)

30    IF (I.GT.NUMWDS) GO TO 40
      IF (I.GT.MAX) GO TO 50
      WRITE(1,31)
31    FORMAT(* *)
      WRITE(1,32) (IADD1(WORD),MEM(J),IBLANK(MEM(J)),J=I,MAX)
32    FORMAT((3(* *,O2,O22,2X,A10,3X)))
      IF (IND.EQ.0) RETURN

40    WRITE(1,41) MARK(ICODE)
41    FORMAT(*0*,R3)
      RETURN

50    WRITE(1,51)
51    FORMAT(*0THIS PRU CONTAINS ONLY ZEROS.*)
```

```
      RETURN
      END

      INTEGER FUNCTION IBLANK(N)
      IBLANK=N
      DO 10 I=1,10
      IBLANK=SHIFT(IBLANK,5)
      IF ((IBLANK.AND.770).EQ.0) IBLANK=IBLANK.OR.558
10    CONTINUE
      RETURN
      END

      INTEGER FUNCTION IADD1(N)
      IADD1=N
      N=N+1
      RETURN
      END
```

```
      SUBROUTINE RBTCHK
      IMPLICIT INTEGER(A-Z)
      COMMON/IFET/IBUFF(2049),RTCODE,NWDS,LEVEL,FILE
      COMMON RBRTBL(3232,8),AREA(3),ENTRY(3),NUMTBL
      COMMON/RBVAR/MCOUNT,WD,I,IERR
      COMMON/RBPORD/MORD(40),NORD(40)

C     IN THIS ROUTINE IBUFF CONTAINS WORDS OF THE RBTC. EACH PRU CAN
C     CONTAIN ONLY ONE RBTC ENTRY. THIS ROUTINE EXAMINES THE ENTRIES.
C     WHEN AN ERROR IS DETECTED A MESSAGE IS PRINTED ON FILE 2. ONLY
C     MCOUNT MESSAGES ARE WRITTEN ON FILE 2. THEN ANALYSIS OF THE RBTC
C     IS TERMINATED.

C     WD = THE NUMBER OF WORDS IN THE RBTC THAT HAVE BEEN PROCESSED
C     I = THE INDEX OF THE WORD IN IBUFF TO BE EXAMINED NEXT

C     THE RBRTBL ARRAY CORRESPONDS TO THE RBR BIT TABLES USED BY THE
C     SYSTEM. HOWEVER, INSTEAD OF REFERENCING AN RB BY A SINGLE BIT (AS
C     IS DONE IN THE BIT TABLES), 20 BITS ARE USED. THE J-TH COLUMN OF
C     THIS ARRAY CONTAINS THE 20 BIT FIELDS FOR THREE RBR ORDINALS.

C     NUMTBL IS THE LARGEST RBR ORDINAL USED. IT CANNOT EXCEED 39.

C     IERR = THE NUMBER OF ERRORS FOUND THAT ARE NOT CONFLICTS

C     AT THE BEGINNING OF A PRU L WILL NORMALLY BE 0. IF L IS NOT 0
C     THEN AN RBTC ENTRY THAT TAKES MORE THAN A PRU IS BEING PROCESSED.
C     IN THIS CASE L IS THE NUMBER OF WORDS IN THE ENTRY THAT HAVE NOT
C     YET BEEN CHECKED.

      L=0
      IZERO=0
1     IF (NWDS.GT.0) GO TO 4
      WDI=WD
      GO TO 11
2     IF (RTCODE.EQ.3) RETURN
      WD=WD+64
3     I=1
      CALL TEST
      GO TO 1
4     NUM=NWDS/64
      NUM=NUM*64
      IPRU=I
      WDPRU=WD

C                 PROCESSING A PRU

10    IF (I.LT.NUM) GO TO 20
      IF (NWDS.EQ.2048) GO TO 3
      NUM=NWDS-NUM
      WDI=WD+NUM
11    MSGE=1
      WRITE(2,12) MSGE,WDI,RTCODE,L,IZERO
12    FORMAT(5O20)
      L=0
      IF (RTCODE.NE.3) IERR=IERR+1
      MCOUNT=MCOUNT-1
```

```
      IF (MCOJNT.EQ.0) RETURN
      GO TO 2

20    W=IBUFF(I)
      IF (L.NE.0) GO TO 300
      IF (W.EQ.0) GO TO 501
      IF ((W.NE.2).AND.(W.NE.228)) GO TO 400
      I=I+1
      WD=WD+1

C             EXAMINING AN RBTC ENTRY

100   W=IBUFF(I).AND.777777777777700008
      IF (W.NE.777777772022403000008) GO TO 400
      I=I+10
      WD=WD+10
      L=IBUFF(I).AND.77770000B
      L=SHIFT(L,-12)
      K=IBUFF(I+1).AND.7777000000008
      K=SHIFT(K,-24)
      IF (L.EQ.(K+12)) GO TO 500
      IND=(L-K).AND.1
      IF ((IND.EQ.1).OR.(L.LT.(K+14)).OR.(K.LT.11).OR.(K.GT.48))
     X     GO TO 402
      IF (L.LE.62) GO TO 110
      MSGE=4
      WRITE(2,12) MSGE,WD,L,IZERO,IZERO
      MCOUNT=MCOUNT-1
      IF (MCOUNT.EQ.0) RETURN

110   L=L-K-12
      I=I+K+2
      WD=WD+K+2
      MAX=(50-K)/2
      ENTRY(3)=SHIFT(WD,-5)
      ENTRY(2)=SHIFT(ENTRY(3),20)
      ENTRY(1)=SHIFT(ENTRY(2),20)

C        READING THE RB CHAIN IN THE ENTRY

200   DO 233 JJ=1,MAX
      LINK=IBUFF(I).AND.777700000000000000008
      W=SHIFT(IBUFF(I),24)
      IND=W.AND.7
      ORD=W.AND.7770B
      ORD=SHIFT(ORD,-3)
      IF (ORD.LE.NUMTBL) GO TO 201
      IF ((ORD.NE.777B).OR.(L.NE.2).OR.(LINK.NE.0)) GO TO 403
      RB=IBUFF(I+1).AND.7777B
      IF ((IND.NE.7).OR.(RB.NE.0)) GO TO 403
      GO TO 500
201   M=MORD(ORD+1)
      N=NORD(ORD+1)
      IF (N.EQ.0) GO TO 403

C     HERE M=1,2,3 AND N=1,2,...,8.
```

```
      IF (IND.EQ.0) GO TO 210
      IF (IND.EQ.3) GO TO 220
      K=WD.AND.7777B
      PRU=SHIFT(WD,-6)
      RB=PRU/56
      PRU=PRU-RB*56
      RB=RB+1
      PRINT 202,IBUFF(I),IBUFF(I+1)
202   FORMAT(*0THE RB BYTE INDICATOR (BITS 36-38 OF THE FIRST WORD) OF T
     XHE RB CHAIN*/* WORD PAIR*,2022,*  IS NOT 0 OR 3,*)
      PRINT 203,IND,RB,PRU,K
203   FORMAT(* BUT *,01,*. THIS WORD PAIR BEGINS AT RB *,04,* PRU *,02,
     X* WORD *,02,*.*)
      IF (IND-3) 211,211,212

210   K=1
      IND=5
      GO TO 221
211   K=1
      W=SHIFT(W,IND*12)
      IND=IND+5
      GO TO 221
212   K=2
      I=I+1
      WD=WD+1
      W=SHIFT(IBUFF(I),(IND-3)*12)
      GO TO 221

220   K=2
      I=I+1
      WD=WD+1
      W=IBUFF(I)
221   DO 223 J=IND,7
      W=SHIFT(W,12)
      RB=W.AND.77777
      IF (RB.EQ.0) GO TO (224,230),K
      IF (RB.GT.3232) GO TO 404
      INFO=AREA(M).AND.RBRTBL(RB,N)
      IF (INFO.EQ.0) GO TO 222
      MSGF=102
      INFO=SHIFT(INFO,20*M)
      WRITE(2,12) MSGF,WD,INFO,RB,ORD
      MCOUNT=MCOUNT-1
      IF (MCOUNT.EQ.0) RETURN
      GO TO 223
222   RBRTBL(RB,N)=ENTRY(M).OR.RBRTBL(RB,N)
223   CONTINUE
      IND=3
      GO TO (220,230),K
224   I=I+1
      WD=WD+1

230   I=I+1
      WD=WD+1
      L=L-2
      IF (L) 232,231
231   IF (LINK) 405,501
```

```
232   IF (LINK) 233,405
233   CONTINUE
      GO TO 501

C       PROCESSING ENTRIES THAT TAKE MORE THAN A PRU

300   IF ((W.NE.2).AND.(W.NE.22B)) GO TO 400
      I=I+1
      WD=WD+1
      W=IBUFF(I).AND.777777777777700008
      IF (W.EQ.7777777722022403000008) GO TO 400
      MAX=31
      ENTRY(3)=SHIFT(WD,-6)
      ENTRY(2)=SHIFT(ENTRY(3),20)
      ENTRY(1)=SHIFT(ENTRY(2),20)
      GO TO 200

C       MESSAGES THAT TERMINATE THE ANALYSIS OF A PRU

400   MSGE=2
401   WRITE(2,12) MSGE,WD,L,IZERO,IZERO
      IERR=IERR+1
      MCOUNT=MCOUNT-1
      IF (MCOUNT.EQ.0) RETURN
      GO TO 500
402   MSGE=3
      WRITE(2,12) MSGE,WD,L,K,IZERO
      IERR=IERR+1
      MCOUNT=MCOUNT-1
      IF (MCOUNT.EQ.0) RETURN
      GO TO 500
403   MSGE=100
      WRITE(2,12) MSGE,WD,IZERO,IZERO,IZERO
      IERR=IERR+1
      MCOUNT=MCOUNT-1
      IF (MCOUNT.EQ.0) RETURN
      GO TO 500
404   MSGE=101
      WRITE(2,12) MSGE,WD,RB,IZERO,IZERO
      IERR=IERR+1
      MCOUNT=MCOUNT-1
      IF (MCOUNT.EQ.0) RETURN
      GO TO 500
405   MSGE=103
      GO TO 401

C             SETUP FOR THE NEXT PRU

500   L=0
501   IPRU=IPRU+64
      WDPRU=WDPRU+64
      I=IPRU
      WD=WDPRU
      GO TO 10
      END
```

```
      SUBROUTINE INFOCON(MCOUNT)
      IMPLICIT INTEGER(A-Z)
      COMMON RBRTBL(3232,8),AREA(3),ENTRY(3),NUMTBL
      EQUIVALENCE (CON(1),RBRTBL(1,1)),(CONPRU(1),CON(20001))
      INTEGER CON(24000),CONPRU(4000)

C     THIS ROUTINE COLLECTS TOGETHER THE CONFLICTS FOUND BY RBTCHK THAT
C     DO NOT INVOLVE FLAWED RBS OR PBS IN THE RB CHAINS FOR THE PFD AND
C     PFC. THESE CONFLICTS ARE STORED IN THE ARRAY CON. CON HAS TWO
C     SECTIONS. THE FIRST SECTION CONSISTS OF THE FIRST 20000 WORDS OF
C     CON. THE SECOND SECTION, WHICH IS THE SUBARRAY CONPRU, BEGINS AT
C     THE 20001-ST WORD OF CON AND CONTAINS 4000 WORDS.

C     THE FIRST SECTION OF CON CONTAINS THE CONFLICT ENTRIES. THE FIRST
C     TWO WORDS OF AN ENTRY CONTAIN THE RBR ORDINAL (ORD) AND THE DISK
C     RB NUMBER (RB) OF AN RB THAT IS IN CONFLICT. THE FORMAT OF THE
C     ENTRY IS ORD,RB,N,WD1,WD2,...,WDN. THUS THE ENTRY CONTAINS N+3
C     WORDS. THE WORD WD1 CONTAINS THE PRU NUMBER (SHIFTED TO THE LEFT
C     6 BITS) THAT WAS STORED IN THE RBRTBL ARRAY FOR THIS RB. THE N-1
C     WORDS WD2,...,WDN CONTAIN THE WORD NUMBERS OF THE WORDS IN THE
C     PFC THAT SUBSEQUENTLY TRIED TO REFERENCE THIS RB (WHEN IT WAS
C     ALREADY KNOWN TO BE IN USE).

C     AN ENTRY IN THE FIRST SECTION OF CON MUST CONTAIN AT LEAST FIVE
C     WORDS. EACH ENTRY REFERS TO A DISTINCT RB IN CONFLICT. THE ENTRIES
C     ARE STORED SEQUENTIALLY, BEING ORDERED BY THE VALUES OF ORD AND
C     RB. SINCE ONLY 20000 WORDS IS BEING ALLOTTED FOR THIS SECTION,
C     THE SECTION CAN CONTAIN AT MOST 4000 ENTRIES. THIS LIMITS THE
C     NUMBER MCOUNT OF MESSAGES THAT CAN BE ISSUED BY RBTCHK TO 4000.
C     (THE PARAMETER MCOUNT IS SET AT THE BEGINNING OF THE MAIN PROGRAM
C     PBTC.)

C     THE SECOND SECTION OF CON, NAMELY THE SUBARRAY CONPRU, CONTAINS
C     THE NUMBERS OF THE PRUS IN THE PBTC THAT FIRST REFERENCED THE RBS
C     IN CONFLICT.

      MAX=0
      MRFR=0
      MPFC=0
      MFLAWS=0
      DO 1 I=1,24000
1     CON(I)=0

      DO 113 I=1,MCOUNT
      READ(2,10) MSGE,WD,INFO,RB,ORD
10    FORMAT(5020)
      IF (MSGE.NE.102) GO TO 113
      W=SHIFT(INFO,-16)+1
      GO TO (14,11,12,13,13),W
11    MPFC=MPFC+1
      GO TO 113
12    MRFR=MRFR+1
      GO TO 113
13    MFLAWS=MFLAWS+1
      GO TO 113
14    MAX=MAX+1
```

```
      J=0
20    J=J+1
      IF (CONPRU(J).NE.0) GO TO 21
      CONPRU(J)=INFO
      GO TO 100
21    IF (CONPRU(J)-INFO) 20,100,22
22    CALL ASHIFT(CONPRU(J),MAX-J+1,1)
      CONPRU(J)=INFO

100   J=1
101   IF (CON(J+1)) 103,102
102   CON(J)=ORD
      CON(J+1)=RB
      CON(J+2)=2
      CON(J+3)=SHIFT(INFO,6)
      CON(J+4)=WD
      GO TO 113
103   IF (CON(J)-ORD) 110,104,111
104   IF (CON(J+1)-RB) 110,112,111

110   J=J+3+CON(J+2)
      GO TO 101
111   CALL ASHIFT(CON(J),5*MAX-J+1,5)
      GO TO 102
112   N=CON(J+2)
      CON(J+2)=N+1
      J=J+3+N
      CALL ASHIFT(CON(J),5*MAX-J+1,1)
      CON(J)=WD
113   CONTINUE

      REWIND 2
      M=MAX+MPFD+MPFC+MFLAWS
      IF (M.NE.0) GO TO 210
      CALL REMARK(19HNO CONFLICTS FOUND.)
      PRINT 200
200   FORMAT(*0NO CONFLICTS FOUND.*)
      RETURN

210   CALL DISPLA(40HNUMBER OF CONFLICTS INVOLVING FLAWS =    ,MFLAWS)
      CALL DISPLA(40HNUMBER OF CONFLICTS INVOLVING PFD RBS = ,MPFD)
      CALL DISPLA(40HNUMBER OF CONFLICTS INVOLVING PFC RBS = ,MPFC)
      CALL DISPLA(40HNUMBER OF OTHER CONFLICTS =             ,MAX)
      CALL DISPLA(40HTOTAL NUMBER OF CONFLICTS =             ,M)
      PRINT 211,MFLAWS,MPFD,MPFC,MAX,M
211   FORMAT(*0NUMBER OF CONFLICTS INVOLVING FLAWED RBS = *,I4,* (DECIMA
     1L).*/* NUMBER OF CONFLICTS INVOLVING PFD RBS = *,I4,* (DECIMAL).*/
     2* NUMBER OF CONFLICTS INVOLVING PFC RBS = *,I4,* (DECIMAL).*
     3* NUMBER OF OTHER CONFLICTS = *,I4,* (DECIMAL).*/
     4* TOTAL NUMBER OF CONFLICTS = *,I4,* (DECIMAL).*)
      RETURN
      END

      SUBROUTINE ASHIFT(A,N,K)
      INTEGER A(N)

C     HERE IT IS ASSUMED THAT K IS A POSITIVE INTEGER EQUAL TO OR LESS
```

```
C       THAN N. THE ROUTINE SHISTS THE DATA IN A(1),...,A(N-K) K WORDS TO
C       THE RIGHT.

        IF (K.EQ.N) RETURN
        I=N
        M=N-K
        DO 10 J=1,M
        A(I)=A(I-K)
10      I=I-1
        RETURN
        END
```

```
      SUBROUTINE GETPRU(PRU,I)
      IMPLICIT INTEGER(A-Z)
      COMMON/IFET/IBUFF(2049),RTCODE,NWDS,LEVEL,FILE
      COMMON/TVAR/IBUFF1(260),PRU0,P,DLIMIT

C     PRU = THE NUMBER OF THE PRU THAT WE ARE LOOKING FOR
C     PRU0 = THE NUMBER OF THE FIRST PRU IN IBUFF
C     P = THE NUMBER OF PRUS THAT WE WISH TO SAVE

C     WHEN THIS ROUTINE TERMINATES IBUFF WILL CONTAIN THE PRU THAT IS
C     WANTED. THIS PRU WILL START AT THE I-TH WORD OF IBUFF. (IF I IS
C     GREATER THAN NWDS THEN THE PRU CONTAINS NO WORDS. IN THIS CASE
C     THE PRU IS TERMINATED BY AN EOR OR EOF MARK.) ALSO SEVERAL PRUS
C     IMMEDIATELY PRECEDING THE FIRST PRU (OF IBUFF) IN THE PFC MAY
C     BE SAVED.

C     THE PRUS THAT ARE TO BE SAVED ARE INSERTED INTO IBUFF1. A PRU
C     ENTRY IN IBUFF1 CONSISTS OF 65 WORDS. THE FIRST WORD OF AN ENTRY
C     CONTAINS THE NUMBER N OF WORDS OF A PRU (N=0,1,...,64). IF THE
C     PRU IS FULL THEN THE REMAINING 64 WORDS OF THE ENTRY CONTAIN
C     THE 64 WORDS OF THE PRU. OTHERWISE, IF N IS LESS THAN 64, THEN
C     THE NEXT N WORDS OF THE ENTRY CONTAIN THE WORDS OF THE PRU. ALSO
C     THE (N+2)-ND WORD OF THE ENTRY CONTAINS THE RTCODE FOR THE EOR
C     OR EOF MARK THAT TERMINATES THE PRU.

C     NUM = THE TOTAL NUMBER OF PRUS IN IBUFF
C     M = THE NUMBER OF PRUS IN IBUFF THAT ARE TO BE SAVED

      GO TO 11

10    CALL TEST
11    NUM=32
      IF (NWDS.NE.2048) NUM=NWDS/64+1
      L=PRU0+NUM
      IF (PRU.LT.L) GO TO 40
      PRU0=L
      IF (PRU.GE.(PRU0+P)) GO TO 10
      M=MIN0(NUM,P)
      CALL ASHIFT(IBUFF1(1),P*65,M*65)
      IF (NWDS.NE.2048) GO TO 20
      I=1
      K=2048
      GO TO 30

20    K=(NUM-1)*64
      N=NWDS-K
      IBUFF1(1)=N
      IBUFF1(N+2)=RTCODE
      IF (N.EQ.0) GO TO 22
      DO 21 JJ=1,N
21    IBUFF1(JJ+1)=IBUFF(K+JJ)
22    IF (M.EQ.1) GO TO 10
      M=M-1
      I=66

30    DO 32 II=1,M
      K=K-64
```

```
      J=K+1
      IBUFF1(I)=64
      I=I+1
      DO 31 JJ=1,64
      IBUFF1(I)=IBUFF(J)
      I=I+1
31    J=J+1
32    CONTINUE
      GO TO 10

40    I=1+(PRU-PRU0)*64
      RETURN
      END

      SUBROUTINE NXTPRU(PRU,I,IND)
      IMPLICIT INTEGER(A-Z)
      COMMON/IFET/IBUFF(2049),RTCODE,NWDS,LEVEL,FILE
      COMMON/CHNVAR/PFDMRK,PFDRBS,PFDCHN(64),PFCMRK,PFCRBS,PFCCHN(320),
     X              MARK(3)

      PRU=PRU+1
      CALL GETPRU(PRU,I)
      IRB=1+PRU/56
      K=PFCMRK
      IF (IRB.GT.PFCRBS) K=1
      CALL PRTPRU(I,PRU,IBUFF(1),NWDS,RTCODE,K,IND)
      RETURN
      END
```

```
      SUBROUTINE REPORT
      IMPLICIT INTEGER(A-Z)
      COMMON/IFET/IBUFF(2049),RTCODE,NWDS,LEVEL,FILE
      COMMON/TVAR/IBUFF1(260),PRU0,P,DLIMIT
      COMMON/CHNVAR/PFCMRK,PFCRBS,PFCCHN(64),PFCMRK,PFCRBS,PFCCHN(320),
     X              MARK(3)
      COMMON RBRTBL(3232,8),AREA(3),ENTRY(3),NUMTBL
      EQUIVALENCE (CON(1),RBRTBL(1,1)),(CONPRU(1),CON(20001))
      INTEGER CON(24000),CONPRU(4000)
      COMMON/RPRT/DMPNUM,FNAME(4),CYCLE,DATE,ID

C     THIS ROUTINE REPORTS THE RESULTS OF THE ANALYSIS DONE BY THE
C     ROUTINE RBTCHK. IF THERE ARE ERRORS IN A PRU IN THE RBTC THEN
C     MESSAGES ARE PRINTED CONCERNING THESE ERRORS AND A BRIEF DUMP
C     IS GIVEN. THE DUMP WILL CONTAIN THE PPU IN QUESTION, PRECEDED
C     NORMALLY BY P PRUS AND FOLLOWED NORMALLY BY Q PRUS. A MAXIMUM
C     OF DLIMIT SUCH DUMPS ARE GIVEN.

C     THE VARIABLES P,Q,DLIMIT ARE ASSIGNED VALUES IN THIS ROUTINE.
C     IT IS ASSUMED THAT P AND Q ARE GREATER THAN OR EQUAL TO 2.
C     DLIMIT MAY BE ANY NONNEGATIVE INTEGER. THE DIMENSION OF THE
C     ARRAY IBUFF1 MUST BE EQUAL TO OR GREATER THAN (P+1)*65. FOR
C     INFORMATION ON THE ARRAY IBUFF1 AND THE VARIABLE PRU0 SEE THE
C     GETPRU ROUTINE.

C     THE VARIABLE CARRY WILL BE 0 IF THE RBTC ENTRY BEING REPORTED
C     ON DOES NOT TAKE MORE THAN A PRU. OTHERWISE CARRY WILL HAVE
C     THE VALUE 1.

C     THE VARIABLE SIG IS NORMALLY 0. HOWEVER, IF IT BECOMES NONZERO
C     THEN THERE IS EITHER A BUG IN THE PROGRAM OR THE SYSTEM HAS BEEN
C     SO BADLY DAMAGED THAT THE PROGRAM CAN NO LONGER RUN PROPERLY. IN
C     THIS CASE THE JOB MUST BE ABORTED.

      P=2
      Q=2
      DLIMIT=150

      PRU0=0
      CALL TEST
      T=1
      DMPNUM=0
      CARRY=0
      SIG=0

      READ(2,1) MSGE,WD,INFO,RB,OFF
1     FORMAT(5020)
      IF (MSGE.EQ.0) RETURN
      WRITE(10,2)
2     FORMAT(*1*,T35,*SUMMARY*)
      GO TO 10

C                 CONTROL SECTION

5     READ(2,1) MSGE,WD,INFO,RB,OFF
10    IF (MSGE.EQ.0) RETURN
      PRU=SHIFT(WD,-6)
```

```
      IF ((T.LE.4000).AND.(CONPRU(T).NE.0)) PRU=MIN0(PRU,CONPRU(T))
      CALL GETPRU(PRU,I)
      DMPNUM=DMPNUM+1
      IF (DMPNUM.GT.DLIMIT) GO TO 13
      IRB=PRU/56
      IPRU=PRU-IRB*56
      IRB=IRB+1
      WRITE(1,11) IRB,IPRU,PRU,DMPNUM
11    FORMAT(*1*,T38,*RB *,O4,* PRU *,O2,* (PFDUMP PRU *,I5.5,* (DECIMAL
     X))*,T100,*DUMP *,O4/)
      WRITE(10,12) IRB,IPRU,DMPNUM
12    FORMAT(11H0**********/*0*,T29,*RB *,O4,* PRU *,O2,T59,*DUMP *,O4/)
13    IF ((PRU.EQ.SHIFT(WD,-6)).AND.(MSGE.LT.100))
     X     GO TO (110,120,122,100),MSGE
      IF (NWDS.LT.(I+63)) GO TO 300
      W=IBUFF(I+1).AND.7777777777777770000B
      IF (W.NE.7777777722022403000OB) GO TO 30
      CARRY=0

20    ID=IBUFF(I+2).AND.7777777777777777000B
      ID=ID.OR.55B
      ID=SHIFT(ID,54)
      DO 21 JJ=1,4
21    FNAME(JJ)=IBUFF(I+2+JJ)
      CYCLE=SHIFT(IBUFF(I+7),12)
      CYCLE=CYCLE.AND.77777B
      DATE=SHIFT(IBUFF(I+8),18)
      DATE=DATE.AND.7777777B
22    IF ((T.GT.4000).OR.(PRU.NE.CONPRU(T))) GO TO 23
      CALL GETCON(I,PRU,SIG)
      IF (SIG.EQ.1) GO TO 310
      T=T+1
23    IF (PRU.NE.SHIFT(WD,-6)) GO TO 40
      K=MSGE-99
      GO TO (160,162,130,170),K

30    IF (CARRY.EQ.0) ID=0
      GO TO 22

40    IF (DMPNUM.LE.DLIMIT) GO TO 200
      PRU=PRU+1
      CALL GETPRU(PRU,I)
41    IF (CARRY.EQ.0) GO TO 10
      IF (MSGE.EQ.0) RETURN
42    IF ((PRU.EQ.SHIFT(WD,-6)).OR.((T.LE.4000).AND.(PRU.EQ.CONPRU(T))))
     X     GO TO 10
      IF (NWDS.GE.(I+63)) GO TO 43
      CARRY=0
      GO TO 10
43    W=IBUFF(I+1).AND.7777777777777770000B
      IF (W.NE.7777777722022403000OB) GO TO 44
      CARRY=0
      GO TO 10
44    PRU=PRU+1
      CALL GETPRU(PRU,I)
      GO TO 42
```

```
C                    MESSAGE PROCESSOR

100   IF (OMPNUM.GT.OLIMIT) GO TO 102
      WRITE(1,101)
      WRITE(10,101)
101   FORMAT(*0THE RBTC ENTRY BEGINNING IN THIS PRU IS LONGER THAN A PRU
     X IN LENGTH.*)
102   READ(2,1) MSGE,WD,INFO,RB,ORD
      CARRY=1
      GO TO 20

110   IF (NWDS.GE.(I+63)) GO TO 301
      CARRY=0
      IF (OMPNUM.GT.OLIMIT) GO TO 5
      WRITE(1,111) MARK(INFO)
      WRITE(10,111) MARK(INFO)
111   FORMAT(*0THIS PRU CONTAINS AN *,R3,* MARK.*)
      GO TO 200

120   CARRY=0
      IF (OMPNUM.GT.OLIMIT) GO TO 5
      WORD=WD.AND.777B
      WRITE(1,121) WORD
      WRITE(10,121) WORD
121   FORMAT(*0WORD *,O2,* OF THIS PRU IS BAD.*)
      GO TO 200
122   CARRY=0
      IF (OMPNUM.GT.OLIMIT) GO TO 5
      WRITE(1,123) INFO,RB
      WRITE(10,123) INFO,RB
123   FORMAT(*0IN THIS PRU, EITHER THE NUMBER *,O4,* GIVEN FOR THE SIZE
     1OF THE ENTRY*/* IN WORD 13 OR THE NUMBER *,O4,* GIVEN FOR THE BEGI
     2NNING OF THE RB CHAIN*/* IS BAD. IF S IS THE FIRST NUMBER AND K TH
     3E SECOND NUMBER, THEN THE RB*/* CHAIN IS ASSUMED TO BEGIN AT WORD
     4K+15 (OCTAL) AND THE NUMBER OF WORDS*/* IN THE RB CHAIN IS ASSUMED
     5 TO BE S-K-14 (OCTAL). ANALYSIS OF TH S PRU*/* WAS TERMINATED WHEN
     6 THIS ERROR WAS DETECTED.*)
      GO TO 200

130   W=SHIFT(INFO,-16)
      IF (W.NE.0) GO TO (150,150,302,140),W
      CALL NTRYCON(WD,RB,ORD)
      READ(2,1) MSGE,WD,INFO,RB,ORD
      GO TO 23

140   IF (OMPNUM.GT.OLIMIT) GO TO 143
      WRITE(1,141) RB,ORD
      WRITE(10,141) RB,ORD
141   FORMAT(*0WE HAVE A CONFLICT INVOLVING RB *,O4,* OF RBR ORDINAL *,
     XO2,*. THIS RB IS*)
      WORD=WD.AND.777B
      WRITE(1,142) WORD
      WRITE(10,142) WORD
142   FORMAT(* FLAWED, AND HENCE NOT USABLE. IT IS REFERENCED IN WORD *,
     XO2,*.*)
143   READ(2,1) MSGE,WD,INFO,RB,ORD
      GO TO 23
```

```
150   IF (OMPNUM.GT.OLIMIT) GO TO 154
      WRITE(1,141) RB,ORD
      WRITE(10,141) RB,ORD
      WORD=WD.AND.77B
      CALL ADDRESS(INFO,K,IRB,IPRU,IWD)
      WRITE(1,153) WORD,IRB,IPRU,IWD,K
      WRITE(10,153) WORD,IRB,IPRU,IWD,K
153   FORMAT(* REFERENCED IN WORD *,O2,* OF THIS PRU AND IN RB *,O4,
     X* PRU *,O2,* WORD *,O2,* OF THE*/* RB CHAIN FOR THE PR*,R1,*.*)
154   READ(2,1) MSGE,WD,INFO,RB,ORD
      GO TO 23

160   CARRY=0
      IF (OMPNUM.GT.OLIMIT) GO TO 5
      WORD=WD.AND.77B
      WRITE(1,161) WORD
      WRITE(10,161) WORD
161   FORMAT(*0THE RBR ORDINAL IN WORD *,O2,* OF THIS PRU IS BAD. ANALYS
     XIS OF THIS PRU*/* WAS TERMINATED WHEN THIS ERROR WAS DETECTED.*)
      GO TO 200
162   CARRY=0
      IF (OMPNUM.GT.OLIMIT) GO TO 5
      WORD=WD.AND.77B
      WRITE(1,163) INFO,WORD
      WRITE(10,163) INFO,WORD
163   FORMAT(*0THE VALUE *,O4,* IN WORD *,O2,* OF THIS PRU IS BAD. ANALY
     XSIS OF THIS PRU*/* WAS TERMINATED WHEN THIS ERROR WAS DETECTED.*)
      GO TO 200

170   CARRY=0
      IF (OMPNUM.GT.OLIMIT) GO TO 5
      WORD=WD.AND.77B
      WRITE(1,171) WORD
      WRITE(10,171) WORD
171   FORMAT(*0AT THE BEGINNING OF THE ENTRY, EITHER THE NUMBER S GIVEN
     1FOR THE SIZE*/* OF THE ENTRY OR THE NUMBER K GIVEN FOR THE BEGINNI
     2NG OF THE RB CHAIN*/* IS BAD. THE RB CHAIN IS ASSUMED TO BEGIN AT
     3WORD K+15 (OCTAL) OF THE*/* ENTRY AND THE NUMBER OF WORDS IN THE R
     4B CHAIN IS ASSUMED TO BE S-K-14*/* (OCTAL). THIS ERROR WAS NOT DET
     5ECTED UNTIL WORD *,O4,*. THEN ANALYSIS*/* WAS TERMINATED.*)


C                  DUMP PRINTER

200   IF ((NWDS.LT.(I+63)).AND.(RTCODE.EQ.3)) GO TO 202
      PRU=PRU+1
      P=P+1
      CALL GETPRU(PRU,I)
202   WRITE(1,203)
203   FORMAT(11H0**********/)

      IRB=1+PRU/56
      K=PFCMRK
      IF (IRB.GT.PFCRBS) K=1
      PD=0
      J=I-64*P
```

```
      IF (J.GE.1) GO TO 220

      IF (PRU.GE.P) GO TO 210
      J=1
      P0=1+(I-1)/64
      GO TO 221

210   IPRU=PRU-P-1
      P0=P-(I-1)/64
      J=1+(P0-1)*65
      DO 211 JJ=1,P0
      N=IBUFF1(J)
      ICODE=0
      IF (N.NE.64) ICODE=IBUFF1(J+N+1)
      CALL PRTPRU(1,IPRU+JJ,IBUFF1(J+1),N,ICODE,K,IND)
211   J=J-65
      J=1

220   P0=P-P0+1
221   IPRU=PRU-P0
      DO 222 JJ=1,P0
      CALL PRTPRU(J,IPRU+JJ,IBUFF(1),NWDS,RTCODE,K,IND)
222   J=J+64
      P=P-1
      IF ((IND.NE.0).AND.(RTCODE.EQ.3)) RETURN

230   IF ((MSGE.EQ.0).OR.(PRU.LE.SHIFT(WD,-6))) GO TO 240
      READ(2,1) MSGE,WD,INFO,RB,ORD
      GO TO 230

240   DO 243 JJ=2,Q
      IF ((PRU.EQ.SHIFT(WD,-6)).OR.((T.LE.4000).AND.(PRU.EQ.CONPRU(T))))
     X     GO TO 250
      IF (NWDS.GE.(I+63)) GO TO 241
      CARRY=0
      GO TO 242
241   IF (CARRY.EQ.0) GO TO 242
      W=IBUFF(I+1).AND.77777777777777770000B
      IF (W.EQ.77777777220224030000B) CARRY=0
242   CALL NXTPRU(PRU,I,IND)
      IF ((IND.NE.0).AND.(RTCODE.EQ.3)) RETURN
243   CONTINUE
      GO TO 41

250   WRITE(1,251)
251   FORMAT(*0NOTE... FOR THE PRUS IMMEDIATELY FOLLOWING THE LAST PRU I
     XN THIS DUMP*)
      K=DMPNUM+1
      WRITE(1,252) K
252   FORMAT(T10,*SEE THE NEXT DUMP (DUMP *,O4,*).*)
      GO TO 10

C          LOSS OF SYNCHRONIZATION SECTION

300   SIG=2
      GO TO 310
301   SIG=3
```

```
      GO TO 310
302   SIG=4

310   K=47474747B
      IRB=PRU/56
      IPRU=PRU-IRB*56
      IRB=IRB+1
      PRINT 311,K,IRB,IPRU,K
311   FORMAT(*0*,R4,* SYNCHRONIZATION LOST AT RB *,O4,* PRU *,O2,*. THIS
     X IS A FATAL CONDITION. SEE THE NEXT PAGE. *,R4)

      PRINT 320
320   FORMAT(*1THERE IS EITHER A BUG IN THIS PROGRAM, OR THE SYSTEM HAS
     1BEEN SO BADLY*/* DAMAGED THAT THE PROGRAM CAN NO LONGER RUN PROPER
     2LY. IN EITHER CASE*/* THE JOB MUST BE ABORTED. THE EOR,EOF,EDI MAR
     3KS ENCOUNTERED WERE ...*/)
      WRITE(1,321)
      WRITE(10,321)
321   FORMAT(*0THE JOB WAS ABORTED PREMATURELY. FOR FURTHER DETAILS SEE
     XTHE MESSAGES PRECEDING THE SUMMARY AND DUMPS.*)

      REWIND 2
330   READ(2,1) MSGE,WD,INFO,RB,GRD
      IF (MSGE.EQ.0) GO TO 340
      IF (MSGE.NE.1) GO TO 330
      WORD=WD.AND.77B
      PRU=SHIFT(WD,-6)
      IRB=PRU/56
      IPRU=PRU-IRB*56
      IRB=IRB+1
      PRINT 331,IRB,IPRU,WORD,MARK(INFO)
331   FORMAT(* RB *,O4,* PRU *,O2,* WORD *,O2,4X,R3,* MARK.*)
      GO TO 330

340   PRINT 341
341   FORMAT(*0IN THE REPORT ROUTINE THE VALUES OF THE MAJOR VARIABLES A
     XT THE TIME*/* OF THE ABORT ARE ...*/)
      PRINT 342,SIG
342   FORMAT(* SIG = *,I1)
      PRINT 343,I
343   FORMAT(* I = *,I4.4,* (DECIMAL)*)
      PRINT 344,PRUO
344   FORMAT(* PRUO = *,I5.5,* (DECIMAL)*)
      RETURN
      END
```

```
      SUBROUTINE GETCON(IO,PRUNUM,SIG)
      IMPLICIT INTEGER(A-Z)
      COMMON/IFET/IBUFF(2049),RTCODE,NWDS,LEVEL,FILE
      COMMON RBRTBL(3232,8),AREA(3),ENTRY(3),NUMTBL
      EQUIVALENCE (CON(1),RBRTBL(1,1)),(CONPRU(1),CON(20001))
      INTEGER CON(24000),CONPRU(4000)

C     PRUNUM = THE NUMBER OF THE PRU WHICH CONTAINS THE CONFLICT
C     IO = THE INDEX OF THE FIRST WORD OF THE PRU

      J=1
      PRU=SHIFT(PRUNUM,6)

10    IF ((J.GT.19995).OR.(CON(J+1).EQ.0)) RETURN
      IF (CON(J+3).EQ.PRU) GO TO 20
11    J=J+3+CON(J+2)
      GO TO 10

20    W=IBUFF(IO+1).AND.777777777777700008
      IF (W.EQ.7777777722022403000008) GO TO 30
      I=IO+1
      WO=1
      MAX=31
      GO TO 100

30    K=IBUFF(IO+12).AND.7777000000008
      K=SHIFT(K,-24)
      WO=K+13
      I=IO+WO
      MAX=(50-K)/2

100   DO 131 II=1,MAX
      LINK=IBUFF(I).AND.777700000000000000008
      W=SHIFT(IBUFF(I),24)
      IND=W.AND.7
      ORD=W.AND.77708
      ORD=SHIFT(ORD,-3)
      IF (ORD.NE.CON(J)) GO TO 123
      IF (IND.EQ.0) GO TO 110
      IF (IND-3) 111,120,112

110   K=1
      IND=5
      GO TO 121
111   K=1
      W=SHIFT(W,IND*12)
      IND=IND+5
      GO TO 121
112   K=2
      I=I+1
      WO=WO+1
      W=SHIFT(IBUFF(I),(IND-3)*12)
      GO TO 121

120   K=2
      I=I+1
      WO=WO+1
```

```
      W=IBUFF(I)
121   DO 122 JJ=IND,7
      W=SHIFT(W,12)
      RB=W.AND.77777B
      IF (RB.EQ.0) GO TO (123,130),K
      IF (RB.NE.CON(J+1)) GO TO 122
      CON(J+3)=PRU+WD
      CALL NTRYCON(CON(J+3),RB,ORD)
      GO TO 11
122   CONTINUE
      IND=3
      GO TO (120,130),K
123   I=I+1
      WD=WD+1

130   IF (LINK.EQ.0) GO TO 132
      I=I+1
131   WD=WD+1
132   SIG=1
      RETURN
      END

      SUBROUTINE NTRYCON(WD,RB,ORD)
      IMPLICIT INTEGER(A-Z)
      COMMON/TVAR/IBUFF1(260),PRUO,P,DLIMIT
      COMMON/RPRT/DMPNUM,FNAME(4),CYCLE,DATE,ID

      WRITE(3) WD,DMPNUM,FNAME,CYCLE,DATE,ID
      IF (DMPNUM.GT.DLIMIT) RETURN

      WORD=WD.AND.77B
      WRITE(1,10) RB,ORD,WORD
      WRITE(10,10) RB,ORD,WORD
10    FORMAT(*0WE HAVE A CONFLICT INVOLVING RB *,O4,* OF RBR ORDINAL *,
     XO2,*. THIS RB IS*/* REFERENCED IN WORD *,O2,*.*)
      RETURN
      END
```

```
      SUBROUTINE CONTBL
      IMPLICIT INTEGER(A-Z)
      COMMON RBRTBL(3232,8),AREA(3),ENTRY(3),NUMTBL
      EQUIVALENCE (CON(1),RBRTBL(1,1)),(CONPRU(1),CON(20001))
      INTEGER CON(24000),CONPRU(4000)

C     THIS ROUTINE AND ITS SUBROUTINE PRTDATA GIVE A TABLE WHICH CROSS
C     REFERENCES THE CONFLICTS NOT INVOLVING FLAWED RBS OR RBS IN THE
C     RB CHAINS FOR THE PFD AND PFC.

      IF (CON(2).EQ.0) RETURN
      J=1
      PRINT 1
1     FORMAT(*1TABLE OF CONFLICTS NOT INVOLVING FLAWED RBS OR RBS IN THE
     X RB CHAINS FOR THE PFD AND PFC*)

10    PRINT 11,CON(J+1),CON(J)
11    FORMAT(11H0**********/*0THE RB IN CONFLICT IS RB *,O4,* OF RBR ORD
     XINAL *,O2,*. THIS RB IS REFERENCED BY ...*)
      N=CON(J+2)
      J=J+3
      DO 12 JJ=1,N
      CALL PRTDATA(CON(J))
12    J=J+1
      IF ((J.GT.19995).OR.(CON(J+1).EQ.0)) RETURN
      GO TO 10
      END

      SUBROUTINE PRTDATA(WD)
      IMPLICIT INTEGER(A-Z)
      COMMON/TVAR/IBUFF1(250),PRU0,P,GLIMIT
      INTEGER FNAME(4)

      WORD=WD.AND.77B
      PRU=SHIFT(WD,-6)
      PB=PRU/56
      PRU=PRU-PB*56
      PB=PB+1
      PRINT 1,PB,PRU,WORD
1     FORMAT(*0PB *,O4,* PRU *,O2,* WORD *,O2)

10    READ(3) W,UNPNUM,FNAME,CYCLE,DATE,ID
      IF (W.EQ.0) GO TO 42
      IF (W.NE.WD) GO TO 10
      IF (ID.EQ.0) GO TO 43

      PRINT 20
20    FORMAT(* THIS WORD APPEARS IN THE FILE WHOSE NAME, CYCLE, ID, AND
     XCREATION DATE ARE ...*)
      PRINT 21,(IBLANK(FNAME(J)),J=1,4)
21    FORMAT(T7,*NAME = *,4A10)
      PRINT 22,CYCLE
22    FORMAT(T7,*CYCLE = *,I4.4,* (DECIMAL)*)
      PRINT 23,IBLANK(ID)
23    FORMAT(T7,*ID =*,A10)

      N=DATE.AND.7777B
```

```
      IF ((N.GT.0).AND.(N.LE.366)) GO TO 31
      PRINT 30
30    FORMAT(T7,*CREATION DATE ... NOT AVAILABLE*)
      GO TO 40
31    YR=SHIFT(DATE,-9)
      LP=0
      K=YR.AND.3
      IF (K.EQ.0) LP=1
      CALL GETDATE(LP,N,MONTH,DAY)
      PRINT 32,MONTH,DAY,YR
32    FORMAT(T7,*CREATION DATE = *,I2.2,*/*,I2.2,*/*,I2.2)

40    IF (DMPNUM.LE.DLIMIT) PRINT 41,DMPNUM
41    FORMAT(* FOR FURTHER DETAILS SEE DUMP *,O4,*.*)
42    REWIND 3
      RETURN
      END

      SUBROUTINE GETDATE(LP,N,M,D)
      INTEGER D

C     THIS ROUTINE COMPUTES THE MONTH (M) AND THE DAY (D) OF THE MONTH
C     FOR THE N-TH DAY OF THE YEAR. (IF THE YEAR IS A LEAP YEAR THEN N
C     CAN HAVE ANY OF THE VALUES 1,2,...,366.)

C     LP=1 IF THE YEAR IS A LEAP YEAR. OTHERWISE LP=0.

      M=1
      D=N
      IF (D.LE.(59+LP)) GO TO 11

      M=3
      D=D-59-LP
      IF (D.LE.153) GO TO 10
      M=8
      D=D-153

10    K=(D-1)/61
      M=M+2*K
      D=D-61*K
11    IF (D.LE.31) RETURN
      M=M+1
      D=D-31
      RETURN
      END
```

APPENDIX B

DISTRIBUTION

DISTRIBUTION

Defense Documentation Center
Cameron Station
Alexandria, Virginia 22314

Edward O. Minasian
2051-28th Ave.
San Francisco, Calif. 94116

Library of Congress
Washington, D.C. 20540
Attn: Gift and Exchange Division

Lorraine Minor
Code: 1892.3
David W. Taylor Naval Ship Research and Development Center
Bethesda, Maryland 20084

Naval Publications and Printing Services Office
NDW
Washington, D.C. 20390

Neil K. McElroy
Code: WA53
Naval Surface Weapons Center
White Oak Laboratory
Silver Spring, Maryland 20910

User Services
Code: 1892.1
David W. Taylor Naval Ship Research and Development Center
Bethesda, Maryland 20084

Local:

DK-70
DK-74 (45)
DX-21
DX-22
DK-60

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR-3534	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ANALYSIS OF RECORD BLOCK CHAINS FOR THE CDC 6700		5. TYPE OF REPORT & PERIOD COVERED Final Rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Alfred H. Morris, Jr.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS (14) NSWC/DL-TR-3534 (11)		12. REPORT DATE August 1976 (12) 13. NUMBER OF PAGES 83 (41p)
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Record Block Chains for the CDC 6700

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report gives a detailed description of a program, currently in operation at NSWC/DL, that determines if the permanent files are properly stored and catalogued on the CDC 6700 computer. The program serves only in a diagnostic capacity. Its purpose is not to correct errors, but only to find and report them. Such a program is needed since system malfunctions can occur. The documentation is intended primarily for system analysts. The report defines each task that is involved, and examines

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

291 598

the coding that performs the task. Background information is provided when needed. The program is for the SCOPE 3.4.3 operating systems.